

# Pregled transformer familije neuronskih mreža i njihova primjena za detekciju govora mržnje

Luka Utješinović

1/22

Univerzitet Crne Gore

Prirodno-matematički fakultet

Računarske nauke, master studije

## Sadržaj

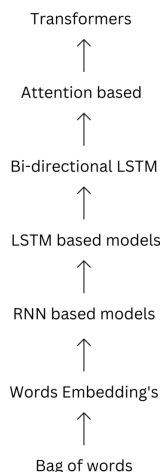
<b>1</b>	<b>Kratak pregled istorije metoda za obradu teksta</b>	<b>2</b>
1.1	Bag of n-grams . . . . .	3
1.2	tf-IDF . . . . .	5
1.3	word2vec . . . . .	6
1.4	word2vec u rekurentnim neuronskim mrežama . . . . .	8
1.5	Algoritmi za tokenizaciju . . . . .	10
<b>2</b>	<b>Transformer familija</b>	<b>13</b>
2.1	Pregled arhitekture . . . . .	14
2.1.1	Pohranjivanje tokena . . . . .	15
2.1.2	Self-attention operacija . . . . .	17
2.1.3	Rezidualne konekcije, normalizacija i MLP . . . . .	19
2.1.4	Specifičnosti dekodirajuće komponente . . . . .	21

2.1.5	Specifičnosti enkoder-dekoder modela . . . . .	23
2.2	Enkoder modeli . . . . .	25
2.2.1	BERT . . . . .	25
2.2.2	RoBERTa . . . . .	29
2.2.3	Sentence BERT . . . . .	31
2.3	Enkoder-dekoder modeli . . . . .	33
2.3.1	T5 . . . . .	33
<b>3</b>	<b>Praktična razmatranja</b>	<b>36</b>
3.1	Podaci, augmentacija i preprocesiranje . . . . .	37
3.2	LoRA - Low Rank Adaptation . . . . .	39
3.3	Izbor metrike za evaluaciju . . . . .	42
3.4	Cross-lingual enkoderi . . . . .	43
3.5	Knowledge distillation . . . . .	44
<b>4</b>	<b>Rezultati</b>	<b>45</b>
	<b>Bibliografija</b>	<b>52</b>

# 1 Kratak pregled istorije metoda za obradu teksta

Kako bi razumijeli transformer familiju, neophodno je imati uvid u algoritme za obradu teksta koji su bili aktuelni u poslednjih 30-40 godina, kako bi uvidjeli nedostatke koji su pokriveni ovom familijom. Najprije ćemo se fokusirati na metode za **vektorizaciju dokumenata** - na koji način se tekstualni dokumenti pohranjuju u modele mašinskog učenja. Ovo je jedan od najosjetljivijih koraka u cjelokupnom procesu modeliranja tekstualnih dokumenata - nepromišljena reprezentacija dokumenta može da dovede do ozbiljnih degradacija u performansama samog algoritma mašinskog učenja. Osvrnućemo se i na **rekurentne neuronske mreže**, može se reći da je transformer familija direktni naslednik ove familije. Iako i danas imaju primjenu, izložićemo razloge zbog kojih se ova familija rijetko koristi u režimima podataka

ogromnog obima i obrade dugih tekstualnih dokumenata.



Slika 1: Sažeti istorijat metoda za obradu teksta. Ova sekcija prati isti graf, u grubim crtama. Izvor: [link](#)

## 1.1 Bag of n-grams

Pretpostavimo da je zadat dokumenat  $d$ . Ne umanjujući opštost (dovoljno je primijeniti adekvatno preprocesiranje), pretpostavimo da su svake dvije uzastopne riječi dokumenta  $d$  razdvojene jednim razmakom. Mi ćemo uvesti bag of  $n$ -grams reprezentaciju dokumenta imajući u vidu da je najmanja gradivna jedinica dokumenta jedna riječ - ovo može da varira u praksi ali glavni principi se ne mijenjaju. U nekoj literaturi se ovaj metod naziva i  $n$ -shingles.

Označimo sa  $l$  dužinu dokumenta  $d$  - broj riječi.  $n$ -gram dokumenta  $d$  predstavlja niz/ $n$ -torku od  $n$  uzastopnih riječi. Na primjer, za dokument  $d = \text{"Alice loves green apples"}$ , jedan mogući  $n$ -gram ovog dokumenta bio bi (Alice, loves). Jasno, dokument dužine  $l$  ima tačno  $l - n + 1$   $n$ -grama. Sa  $d_i^n$  označavamo  $i$ -ti  $n$ -gram dokumenta  $d$ . Sa  $G_n(d)$  označavamo skup svih  $n$ -grama dokumenta  $d$ ,  $G_n(d) = \{d_i^n \mid i = 1, |d| - n + 1\}$ . Broj pojavljivanja  $n$ -grama  $d_i^n$  u dokumentu  $d$  označimo sa  $f(d, d_i^n)$ .

Dalje, neka je  $D = \{d_i\}_{i=1}^N$  korpus dokumenata sa kojim raspolažemo. Skup svih  $n$ -grama koji se javljaju u korpusu  $D$  je  $G_n(D) = \cup_{i=1}^N G_n(d_i)$ ,  $|G_n(D)| = L$ . Izvršimo proizvoljnu enumeraciju elemenata skupa  $G_n(D) = \{x_1, \dots, x_L\}$ . Tada se svaki dokument  $d$  korpusa  $D$  može predstaviti kao vektor dužine  $L$ , u oznaci  $F_n(D, d)$  na sledeći način:

$$d \equiv F_n(D, d) = (f(d, x_1), \dots, f(d, x_L))$$

Moguće je odraditi skaliranje elemenata prethodnog vektora koristeći  $L$ , dužinu dokumenta  $d$  ili neki drugi parametar, ne umanjujući opštost zadržaćemo se na ovoj reprezentaciji. Istorijski gledano, ovo je bila prva prekretnica za vektorizaciju dokumenata, i ovako dobijeni vektori mogu se lako pohraniti u, kako klasične, tako i napredne algoritme mašinskog učenja. Međutim, ovaj metod ima brojnih nedostataka:

- **Redundantnost** - Dužina vektorske reprezentacije je  $L$ , a kao što smo vidjeli  $L$  predstavlja broj jedinstvenih  $n$ -grama koji se pojavljuju unutar čitavog korpusa  $d$ . Imajući u vidu da je dužina samog dokumenta  $|d| = l \ll L$ , mi koristimo značajno više parametara za predstavljanje jednog dokumenta u odnosu na njegovu dužinu, pa je koeficijent iskorišćenosti vrlo loš. Dodatno, kako je  $d$  zapravo samo mali podskup  $n$ -grama unutar skupa svih  $n$ -grama koji se javljaju u korpusu  $D$ , to povlači da je  $F_n(D, d)$  izuzetno rijedak vektor, a to može da utiče loše na pojedine algoritme mašinskog učenja - na primjer za logističku regresiju, prilikom treniranja i obrade dokumenta  $d$ , neće doći do promjene koeficijenata koji odgovaraju 0 koordinatama vektora  $F_n(D, d)$ .
- **Invarijantnost u odnosu na permutacije  $n$ -grama** - Pošto se  $F_n(D, d)$  suštinski konstruiše koristeći statistička svojstva od  $d$  - koliko se puta zadati  $n$ -gram pojavljuje u dokumentu  $d$ , to povlači da je  $F_n(D, d) = F_n(D, \pi(d))$ , gdje je  $\pi(d)$  dobio od  $d$  primijenjujući proizvoljnu permutaciju  $\pi$  nad torkom  $(d_1^n, \dots, d_{|d|-n+1}^n)$ . Na primjer, razmotrimo dva dokumenta,  $d_1 =$

"The cat chased a mouse",  $d_2 =$  "The mouse chased a cat". Za  $n = 1$ , ove dvije rečenice će imati istu bag of  $n$ -grams reprezentaciju, a suštinski se razlikuju, a to bi moglo da ima negative posledice po algoritam mašinskog učenja.

- **Potreba za lematizacijom** - Jedan metod da se redukuje  $L$  u praksi jeste primjena algoritama za lematizaciju. Na primer, ako posmatramo  $n = 1$ , i dva unigrama, "flies", "fly", primijećujemo da semantički vrlo slični. Intuitivno, algoritmi koji se fokusiraju na klasifikaciju teksta treba da uzmu u obzir semantiku samog teksta, u krajnjem slučaju gramatička korektnost ulaznog teksta ne bi trebalo da bude od značaja. U ovakvim situacijama, u praksi se gotovo uvijek primjenjuju algoritmi za **lematizaciju** - svođenje riječi na "korijeni" oblik. U prethodnom primjeru, jedna moguća redukcija bila bi "files"  $\rightarrow$  "fly". Ako se lematizacija uključi kao korak u preprocesiranju, jasno je da se može značajno može redukovati dimenzija rezultujućih vektora, međutim prethodno opisani problemi su i dalje prisutni (redundantnost je ublažena ali invarijantnost u odnosu na permutacija je i dalje prisutna). Dodatno, sami algoritmi sa lematizaciju usložnjavaju preprocesiranje, a neopreznom lematizacijom takođe se mogu narušiti i performanse samog algoritma.

## 1.2 tf-IDF

Radi kompletnosti, kratko se osvrnimo na **tf-IDF - term frequency - Inverse Document Frequency** metodu. Neka je  $t$  proizvoljni  $n$ -gram koji se javlja u korpusu  $D$ , i neka je  $d$  proizvoljan dokument korpusa. Definišemo sledeće funkcije:

$$tf(d, t) = \frac{f(d, t)}{\sum_{x \in G_n(d)} f(d, x)}$$

$$idf(D, t) = \log\left(\frac{|D|}{|\{y \in D \mid t \in G_n(y)\}|}\right)$$

tf-IDF koeficijent za korpus  $D$ , dokument  $d$  i  $n$ -gram  $t$  definiše se na sledeći način:

$$tfidf(d, t, D) = tf(d, t) \times idf(D, t)$$

Analizirajući prethodnu jednakost, vidimo da je  $tfidf$  koeficijent visok ako se  $n$ -gram  $t$  često javlja u dokumentu  $d$ , ali nije previše zastupljen u u korpusu  $D$ . Intuitivno, ovaj koeficijent je korisniji od obične frekvencije  $f(d, t)$  jer uzima u obzir i pojavljivanja zadanog  $n$ -grama unutar čitavog korpusa. Ako je  $G_n(D) = \{x_1, \dots, x_L\}$ , slično se definiše i  $tfidf$  vektorska reprezentacija dokumenta:

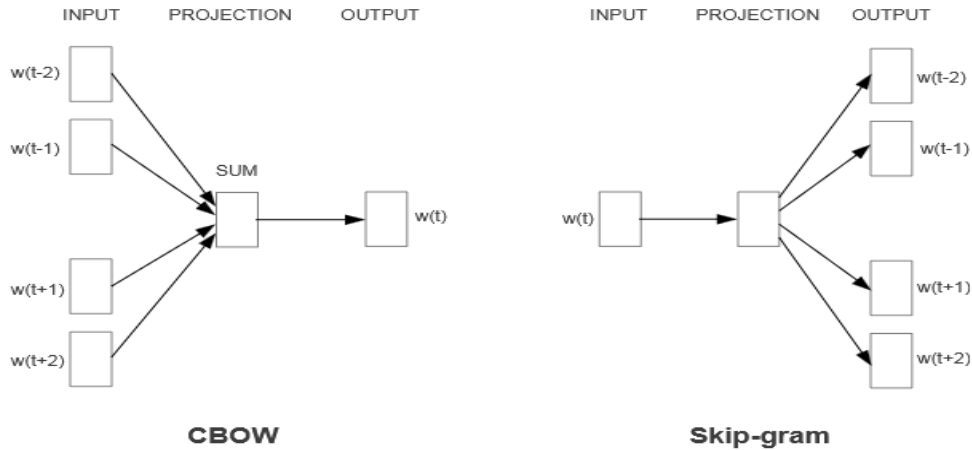
$$d \equiv F'_n(D, d) = (tfidf(d, x_1, D), \dots, tfidf(d, x_L, D))$$

U praksi, ova reprezentacija često dovodi do boljih performansi, u poređenju sa klasičnim bag of  $n$ -grams metodom, ali i dalje su prisutni svi nedostaci koje smo naveli i za bag of  $n$ -grams pristup. Na kraju, navodimo da je ovo osnovna definicija tf-IDF koeficijenta, postoje razne adaptacije/re-skaliranja ovog metoda, ali suština ostaje ista.

### 1.3 word2vec

Kod prethodno opisanih metoda, imali smo generisanje vektorske reprezentacije dokumenata na osnovu statističkih svojstava cjelokupnog korpusa. Pošto su ovi vektori generisani koristeći isključivo statistička svojstva korpusa, na intuitivnom nivou, to znači da pored rešavanja zadanog problema (klasifikacija, autoregresivni language modelling, ...) mi obavezujemo algoritam mašinskog učenja da dodatno uči i semantiku zadatah vektora. Prirodna je pretpostavka da se prethodni problemi ne mogu riješiti ako algoritam na neki način nije enkodirao semantiku zadatah vektora u svojim parametrima, a algoritam je već “opterećen” rešavanjem inicijalnog problema. Iako su prethodna razmatranja isključivo filozofske prirode, motivisali su značajan broj metoda koji se vodi ovom paradigmom - razdvajanje ciljanog problema od razumijevanja semantike vektorskih reprezentacija.

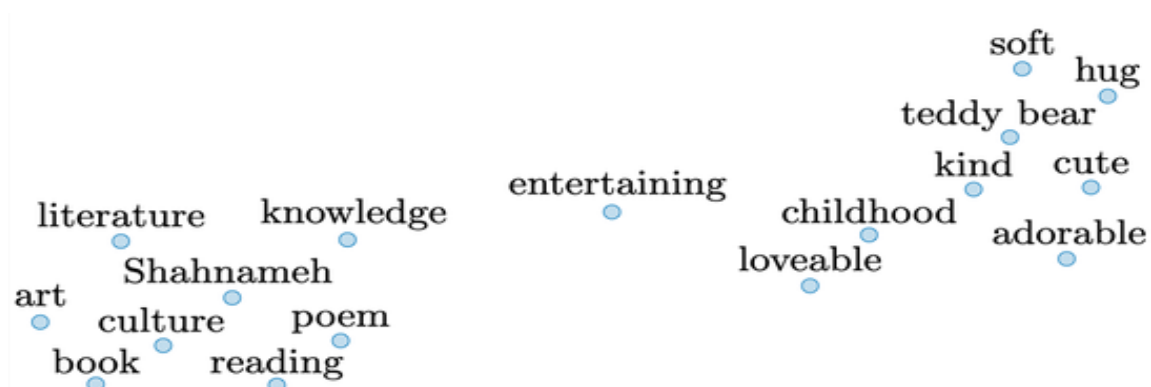
Vjerovatno najpopularniji metod na ovu temu je **word2vec**, [12]. U kratkim crtama ćemo opisati algoritam treniranja. Neka je  $V$  vokabular riječi koji je dobijen iz nekog korpusa  $D$ . Svakoj riječi iz zadatog vokabulara pridružujemo vektor dužine  $d$ , ovo se može predstaviti matricom  $E \in \mathbb{R}^{|V| \times d}$ . Autori su pokazali da  $d = 300$  daje impresivne rezultate. Vektor za riječ  $e_i$  prevodimo u vektor  $s_i$  dimenzionalnosti  $|V|$  linearnom transformacijom  $s_i = He_i, H \in \mathbb{R}^{|V| \times d}$ . Koordinate vektora  $s_i$  mogu da se koriste kao logit koeficijenti za predikciju riječi koje su u neposrednoj blizini trenutne riječi  $w$  - vjerovatnoće dobijamo transformacijom  $\text{softmax}(s_i)$ . Preciznije, za dužinu konteksta  $C$ , uzimamo  $2C + 1$  uzastopnih riječi iz korpusa,  $w_{-C}, \dots, w_{-1}, w_0, w_1, \dots, w_C$ , računamo  $s_0$  i koristimo logit koeficijente za predikciju riječi iz zadatog konteksta - ovim je opisan **Skip-gram** word2vec algoritam. Postoji i continuous bag of words algoritam - **CBOW**, on funkcioniše u "obrnutom" smjeru, predikcija  $w_0$  na osnovu konteksta.



Slika 2: CBOW i Skip-gram word2vec algoritmi. Izvor: slika 1 iz [12]

Rezultat algoritma jestu naučeni vektori  $E$ , transformacija  $H$  se može odbaciti. Iz prethodno izloženog vidimo da se radi o **self-supervised** algoritmu treniranja, tako da je moguće skupiti tekstualne korpusse velikih razmjera i pohraniti ih u model za treniranje, autori su pokazali da se na ovaj način mogu naučiti **semantički bogati** vektori riječi -

semantički slične riječi imaju slične vektore - u odnosu na kosinusnu sličnost. Na primjer, nakon treninga autori demonstriraju da, ako je  $X = E(\textit{France}) - E(\textit{Paris}) + E(\textit{Germany})$ , onda je  $X$  najbližiji vektoru  $E(\textit{Berlin})$ . Očigledna je upotrebnost ovako naučenih vektora: od klasičnih metoda obrade prirodnih jezika kao što je information retrieval, do njihovog pohranjivanja u algoritme mašinskog učenja. U drugom slučaju, algoritmu mašinskog učenja se kao ulaz predaju semantički bogati vektori, tako da je uklonjen problem koji je izložen na početku ove sekcije.



Slika 3: Ilustracija naučenih word2vec reprezentacija, nakon primjene algoritma za redukciju dimenzionalnosti. Uočavamo korelaciju između semantičke i kosinusne sličnosti. Izvor: [19]

Radi kompletnosti, navodimo još jedan algoritam za učenje semantički bogatih vektorskih reprezentacija, **GloVE - Global Vectors for word representations**, [1].

## 1.4 word2vec u rekurentnim neuronskim mrežama

Treba imati u vidu invarijantnost u odnosu na permutaciju prilikom korišćenja ovih vektora. Na primjer, jedan način pohranjivanja ovih vektora u model logističke regresije jeste da se vektori svih riječi trenutnog dokumenta agregiraju u jedan vektor. Pošto je većina agregatnih funkcija invarijantna u odnosu na permutaciju argumenata, to svojstvo se prenosi i na sami algoritam učenja, a prethodno smo opisali zašto ovo



svojstvo nije poželjno. Alternativa je da se koriste **sequence-aware** algoritmi, kao što su **rekurentne neuronske mreže**, i ovaj metod je do skoro bio de-facto standard kada je u pitanju duboko učenje nad tekstualnim podacima. Rekurentne neuronske mreže nisu u fokusu ovog projekta, iako su same po sebi interesantna tema. Sažeti prikaz različitih varijanti ovih modela može se vidjeti na 4, ovdje izlažemo neke praktične nedostatke ove familije, prije svega u cilju motivacije transformer arhitekture:

- **Paralelizacija** - Rekurentno svojstvo ovih neuronskih mreža, po kojem je ova familija dobila ime, leži u sledećoj jednakosti:  $h^t = f(W_{hh}h^{t-1} + W_{xh}x^t + b)$ , gdje je  $h^t$  hidden state za vrijeme  $t$ ,  $x^t$  je ulaz za vrijeme  $t$ ,  $f$  je proizvoljna funkcija aktivacije. Jasno je da je moguće paralelizovati procesiranje više nizova istovremeno (proširenje vektora  $h^t, x^t$ ), ali zbog prethodne relacije nije moguća paralelizacija izračunavanja po vremenskoj dimenziji - da bi izračunali  $h^t$  neophodno je da znamo vrijednost  $h^{t-1}$ .
- **Modeliranje dugih sekvenci** - Kako bi objasnili ovaj problem, posmatrajmo najjednostavniji režim. Neka je  $h^t = [h^t, x^t]$ ,  $b = 0$ ,  $f$  identičko preslikavanje. Onda imamo  $h^t = W_{hh}h^{t-1}$ . Lako se dobija da je  $h^t = W_{hh}^t h^0$ , gdje je  $h^0$  početni hidden state, i u praksi se najčešće inicijalizuje kao konstantni vektor. Ako pretpostavimo da je linearni operator  $W$  dijagonalizibilan, onda je  $h^t = (QD^tQ^T)h^0$ , gdje je  $D$  dijagonalna matrica sopstvenih vrijednosti od  $W$ , a  $Q$  je matrica koja sadrži odgovarajuće sopstvene vektore od  $W$ , po kolonama. Za dovoljno veliko  $t$ , sopstvene vrijednosti koje su veće od 1 po apsolutnoj vrijednosti će da "eksplodiraju", a sopstvene vrijednosti koje su manje od 1 po apsolutnoj vrijednosti će da "implodiraju", što uslovljava **exploding/vanishing gradients** problem - gradijenti koji se koriste prilikom optimizacije postaju nestabilni. Za prvi problem u praksi se često koristi ad-hoc metod, tzv. **gradient clipping** - norma gradijenta se prilikom treniranja drži konstantnom,  $g \leftarrow \gamma \frac{g}{\|g\|}$ , gdje je  $\gamma > 0$  hiperparametar. Drugi problem je

znatno teži, postoje određene specijalizovane arhitekture kao što su **GRU - Gated Recurrent Unit** ili **LSTM - Long-Short Term Memory** koje donekle ublažavaju ovaj problem, ali se često pojavljuje u praksi za dovoljno veliko  $t$ .

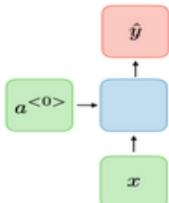
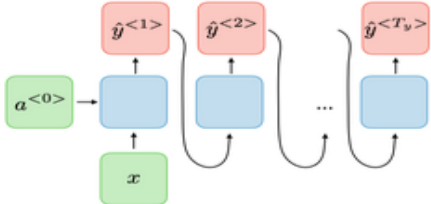
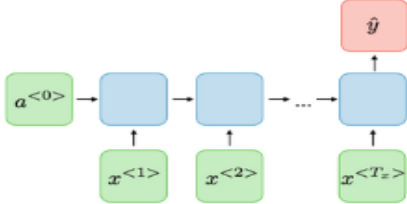
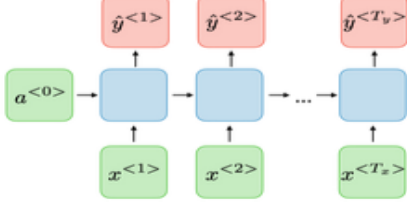
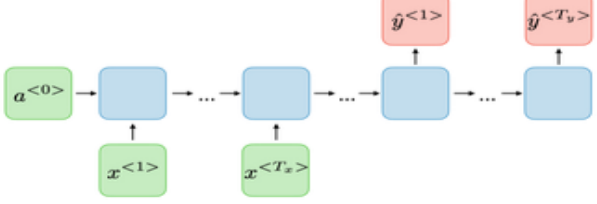
## 1.5 Algoritmi za tokenizaciju

Za razumijevanje transformer familije, potrebno je da razumijemo algoritme za tokenizaciju, pa ih opisujemo u ovoj sekciji. Posmatrajmo **autoregressive language modelling problem**. Neka je  $d = w_1 \dots w_n$  proizvoljan dokument korpusa  $D$ .  $w_i \in V$  mogu biti riječi, pojedinačni karakteri dokumenta  $d$ , ili nešto između. Reći ćemo da su  $w_i$  **tokeni** dokumenta  $d$ . Svaki dokument dopunimo na sledeći način:  $d \leftarrow \langle \text{start} \rangle d \langle \text{end} \rangle$ , gdje su  $\langle \text{start} \rangle$ ,  $\langle \text{end} \rangle$  dva nova, specijalna tokena. Želimo parametrizovanu, kategoričku vjerovatnosnu raspodjelu koja uzima niz proizvoljne dužine  $w_1, \dots, w_k$  i dodjeljuje vjerovatnoće svim mogućim sledećim tokenima  $w_{k+1}$ . Dakle, za dokument  $d$ , imamo:

$$L(\theta; d) = - \sum_{i=2}^n \log(p_{\theta}(w_i \mid w_1 \dots w_{i-1}))$$

Optimizujući  $L(\theta; d)$  nad cijelim korpusom  $D$  dobija se **language model**. Zaista, nakon optimizacije, krenuvši od  $w_1 = \langle \text{start} \rangle$ , uzorkovanjem nad  $p_{\theta}$  dobijamo sledeći najvjerovatniji token, sve dok taj token nije  $\langle \text{end} \rangle$ . Sami algoritam uzorkovanja može da varira (greedy sampling, beam search, top-k sampling, top-p sampling, ...) ali princip ostaje isti.

Postavlja se pitanje kako izvršiti tokenizaciju. Ako uzmemo da je najmanja gradivna jedinica jednog dokumenta jedan karakter, onda je  $V$  predstavljen sa svim alfanumeričkim karakterima. To znači da je će dužina dokumenta biti jednaka broju karaktera, što znači da će nam veliki broj dokumenata ulaznog korpusa biti predstavljen izuzetno

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification
Many-to-many $T_x = T_y$		Name entity recognition
Many-to-many $T_x \neq T_y$		Machine translation

Slika 4: Ilustracija različitih arhitektura rekurentnih neuronskih mreža, na visokom nivou. Za slučaj obrade teksta, kao ulaz bismo ovakvom algoritmu predali naučene word2vec reprezentacije. Izvor: [19]

dugim nizovima tokena, što nije povoljno - već smo vidjeli da pojedini algoritmi ispoljavaju probleme prilikom obrade dugih dokumenata.

U drugoj krajnosti, ako odaberemo da je  $V$  skup svih riječi zadatog jezika, imaćemo da su dokumenti relativno kratki, što je povoljno. Međutim,  $p_\theta(. \mid w_1 \dots w_k)$  je vektor dužine  $V$  za bilo koje  $k$ . Pri tome, velika većina algoritama mašinskog učenja računa vjerovatnoće na sledeći način:

$$p_\theta(. \mid x_1 \dots x_k) = \text{softmax}(g_\theta(w_1, \dots w_k), g_\theta(w_1, \dots w_k)) \in \mathbb{R}^{|V|}$$

Uočavamo primjenu *softmax* transformacije nad vektorom dužine  $|V|$ . Pošto smo odabrali  $V$  kao skup svih riječi nekon jezika (ili neki njegov podskup), za očekivati je  $V$  izuzetno veliko. U praksi nije rijetkost  $|V| \sim 10^6$ . Potpuno izračunavanje ovog izraza zahtjeva ozbiljne resurse, čak i sa moderne standarde, optimizacija ovakvog modela bi trajala predugo. U principu tražimo kompromis: ne želimo da su nam ulazni dokumenti predugi, a istovremeno ne želimo da rezultujući vokabular bude prevelik. Za ove potrebe uvedeni su algoritmi za automatsku tokenizaciju dokumenata, koji koriste statistička svojstva ulaznog korpusa za određivanje vokabulara koji pravi kompromis između prethodne dvije veličine - veličina vokabulara i dužina dokumenta.

Jedan od najpopularnijih algoritama ovog tipa je **BPE - Byte Pair Encoding**. Prvi put je izložen u [16], ali je popularizovan tek nešto kasnije otkrivanjem transformer arhitekture. Ovdje izlažemo **byte-level BPE** algoritam, koji se koristi od strane modela kao što su **GPT-x**, **LLaMA-x**, **RoBERTa**, izložen je u 1.

Još jedna pogodnost byte-level varijante ovog algoritma je što dokument posmatramo kao niz bajtova, tako da nije moguće da se desi **OOV - Out of Vocabulary** problem, što recimo nije slučaj sa prethodno izloženom word2vec metodom.

Radi kompletnosti, navodimo još jedan algoritam za tokenizaciju koji se često koristi u praksi - **SentencePiece**, [9].

Born in London, Turing was raised in southern England. He graduated from King's College, Cambridge, and in 1938, earned a doctorate degree from Princeton University. During World War II, Turing worked for the Government Code and Cypher School at Bletchley Park, Britain's codebreaking centre that produced Ultra intelligence. He led Hut 8, the section responsible for German naval cryptanalysis

Born in London, Turing was raised in southern England. He graduated from King's College, Cambridge, and in 1938, earned a doctorate degree from Princeton University. During World War II, Turing worked for the Government Code and Cypher School at Bletchley Park, Britain's codebreaking centre that produced Ultra intelligence. He led Hut 8, the section responsible for German naval cryptanalysis

59204, 304, 7295, 11, 95530, 574, 9408, 304, 18561, 9635, 13, 1283, 33109, 505, 6342, 596, 9304, 11, 24562, 11, 323, 304, 220, 7285, 23, 11, 15662, 264, 10896, 349, 8547, 505, 50421, 3907, 13, 12220, 4435, 5111, 8105, 11, 95530, 6575, 369, 279, 10423, 6247, 323, 18221, 29182, 6150, 520, 426, 1169, 331, 3258, 5657, 11, 13527, 596, 2082, 37757, 12541, 430, 9124, 29313, 11478, 13, 1283, 6197, 67413, 220, 23, 11, 279, 3857, 8647, 369, 6063, 46398, 14774, 35584

Slika 5: Primjer naučenog BPE vokabulara koji se koristi za model GPT 3.5. Niz Identifikatora tokena je ono što se pohranjuje u algoritam. Ovako dobijen vokabular je kardinalnost  $\sim 5 \times 10^4$ , što je značajno manje u odnosu na  $10^6$ . Izvor: tiktokenizer

## 2 Transformer familija

Može se reći da je otkrivanjem transformer arhitekture došlo do revolucije u svijetu dubokog učenja, najprije polazeći od algoritama za obradu teksta. Iako su rekurentne neuronske mreže davale dobre rezultate za podatke manjeg obima, prethodno smo izložili njihove nedostatke: težina paralelizacije, modeliranje dugih sekvenci i vanishing gradients problem. Ovi problemi su gotovo u potpunosti neutralisani transformer arhitekturom. Dodatno, ispostavlja se da ova familija modela ispoljava vrlo povoljna svojstva u praksi, po pitanju povećanje obima podataka/povećanje broja trening iteracije/povećanje broja parametara modela, što se može vidjeti na slikama 6, 7.

U nastavku ćemo izložiti matematički pregled ove arhitekture. Na kraju, napominjemo da iako su ovi modeli prvenstveno dizajnirani za obradu teksta, mogu da se primjene i nad audio i vizuelnim podacima. Štaviše, u posljednje vrijeme popularan je trend **multimodalnosti** - dizajniraju se modeli koji mogu istovremeno da obrađuju različite

---

**Algorithm 1** byte\_level BPE(D, L)

---

▷ D - trening korpus, dobijen konkatenacijom svih dokumenata. D posmatramo kao UTF-8 enkodirani stream bajtova.  
▷ L - Hiperparametar koji određuje veličinu izlaznog vokabulara  
▷ Prvih 256 bajtova čini inicijalni vokabular

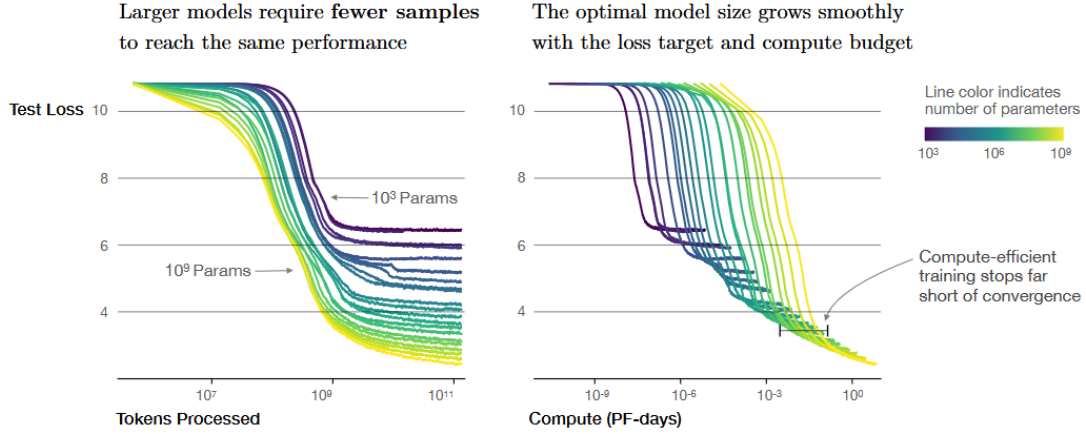
```
1:  $V \leftarrow \{0, \dots, 255\}$ 
2: for  $i = 1, \dots, L - 256$  do
3:   for  $x \in V$  do
4:     for  $y \in V$  do
5:        $f(x, y) = \text{Broj pojavljivanja } (x, y) \text{ u } D$ 
6:     end for
7:   end for
8:    $x^*, y^* = \arg \max_{x, y} (f(x, y))$ 
9:    $z \leftarrow x^*y^*$  novi token
10:   $V \leftarrow V \cup \{z\}$ 
11:  Sva pojavljivanja  $(x^*, y^*)$  u  $D$  zamijeniti sa  $z$ 
12: end for
    return  $V$ 
```

---

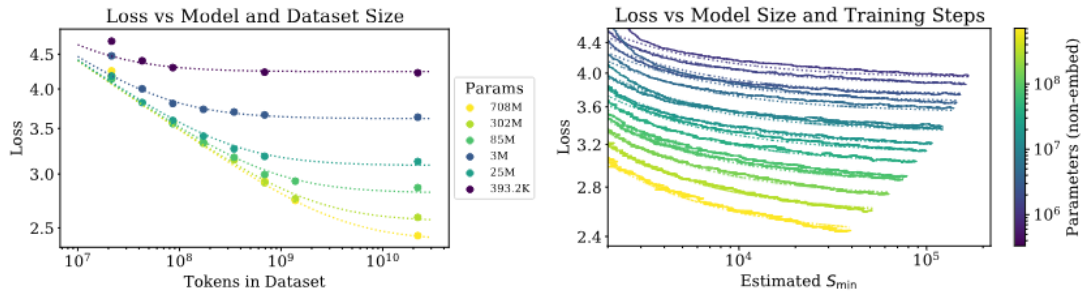
tipove podataka, npr. istovremeno obrađivanje slika i teksta. Kao jednog predstavnika ovakvih modela, navodimo [13].

## 2.1 Pregled arhitekture

Ova arhitektura prvi put je izložena u [22], za rješavanje problema mašinskog prevođenja. Može se napraviti paralela sa 4, **enkoder** model obrađuje rečenicu na izvornom jeziku, i rezultujuću reprezentaciju rečenice predaje **dekoder** modelu koji vrši autoregresivno prevođenje na ciljani jezik. Na slici 9 možemo vidjeti dijagram ove arhitekture. U principu, za rješavanje problema klasifikacije teksta (detekcija govora mržnje), dovoljna je samo enkoder komponenta, međutim moguće je koristiti i pune enkoder-dekoder modele, vidjećemo kako u nastavku. Kao što je i očekivano, vidjećemo da su enkoder i dekoder komponente veoma slične. Primjer problema koji koristi samo dekoder komponentu



Slika 6: Kako dužina trening faze utiče na performanse transformer modela. Izvor: [7]

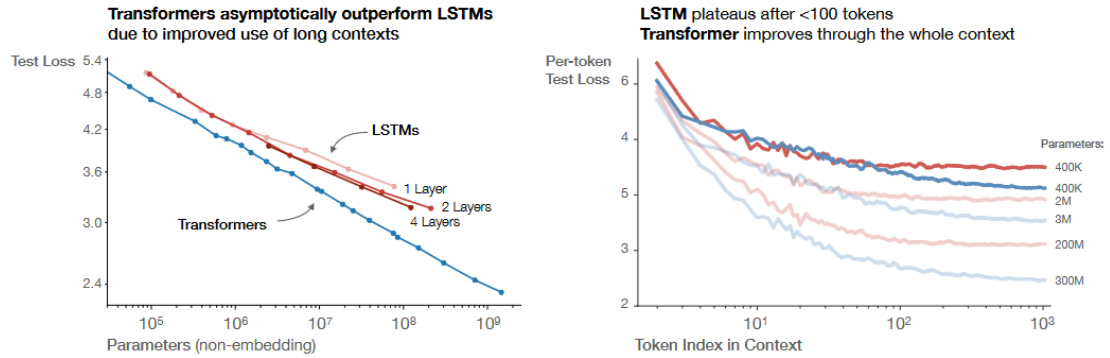


Slika 7: Kako obim trening podataka utiče na performanse transformer model, i poređenje brzine konvergencije za transformer modele različitog obima. Izvor: [7]

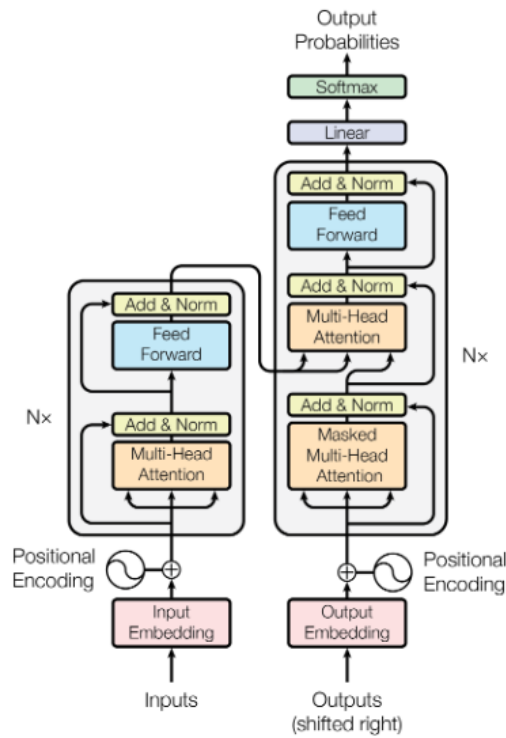
jeste autoregresivni language modelling, chat-bot asistenti kao što su Chat-GPTx, Gemini, DeepSeek-Vx, ...

### 2.1.1 Pohranjivanje tokena

Neka je  $V$  vokabular dobijem primjenom nekog od algoritma za tokenizaciju. Vidjeli smo da se ulazni dokument/rečenica može predstaviti kao niz identifikatora tokena, gdje su ovi identifikatori od 0 do  $|V| - 1$ . Svaki od ovih identifikatora nam služi kao indeks za pronalaženje vek-



Slika 8: Poređenje transformer i LSTM RNN familija. Izvor: [7]



Slika 9: Dijagram transformer arhitekture, na lijevoj strani imamo enkoder, a desnoj dekoder. Izvor: [22]

torske reprezentacije tokena - token sa identifikatorom  $i$  biće predstavljen sa vektorom  $E_i$ , gdje je  $E \in \mathbb{R}^{|V| \times d_{model}}$  matrica koja sadrži ove reprezentacije, a  $E_i$  je  $i$ -ta vrsta ove matrice. Ova matrica se ini-



cijalizuje sa slučajnim vrijednostima, i tretira se kao parametar modela, što znači da će se prilikom treniranja mijenjati - slično kao kod word2vec. Kako bi izbjegli problem invarijantnosti u odnosu na permutaciju tokena, uvode se **poziciona enkodiranja**. Ako je  $T$  dužina dokumenta koju model može da obradi (u literaturi se ovo naziva i **context length**), onda se uvodi matrica  $PE \in \mathbb{R}^{T \times d_{model}}$ , pri čemu je onda interna reprezentacija tokena  $i$  dobijena sa  $E_i + PE_i$ . U principu, dva su osnovna načina za implementaciju ovog mehanizma:

- $PE$  se slučajno inicijalizuje i takođe se tretira kao parametar modela - mijenja se kroz vrijeme.
- Koristeći **statička poziciona enkodiranja**, autori [22] su predložili sledeće:  $PE[pos, 2i] = \sin(pos/10^{\frac{8i}{d_{model}}})$ ,  $PE[pos, 2i + 1] = \cos(pos/10^{\frac{8i}{d_{model}}})$ . Dvije su prednosti ovog pristupa: redukcija broja parametara modela jer se ove vrijednosti tretiraju kao konstante, i ekstrapolacija na duže dokumente  $T' > T$  koji se nisu javljali tokom treniranja, što podrazumijeva jednostavno proširenje  $PE$  matrice po prethodnim formulama.

Jasno je da ako na ulazu imamo tokene  $i, j$  koji se poklapaju, onda je  $E_i = E_j$ , ali je u opštem slučaju  $E_i + PE_i \neq E_j + PE_j$ , čime se izbegava invarijantnost u odnosu na permutaciju tokena - bitno je koji token se nalazi na kojoj poziciji. Napominjemo da je ovo najjednostavniji način implementacija pozicionih enkodiranja, postoje i neke naprednije varijante kao što su **relative positional encodings** i **RoPE- Rotary Positional encodings**, ali ih nećemo obrađivati ovdje.

### 2.1.2 Self-attention operacija

Sledeća operacija je **self-attention**. Pretpostavimo da je  $H \in \mathbb{R}^{T \times d_{model}}$ . Prvo se računaju:

$$Q = HW^Q$$

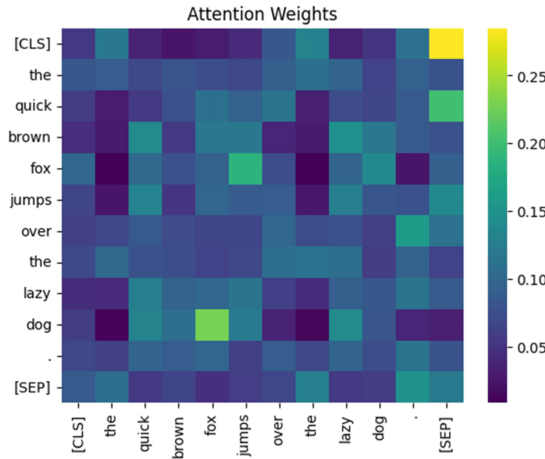
$$K = HW^K$$

$$V = HW^V$$

gdje su  $W^Q, W^K \in \mathbb{R}^{d_{model} \times d_k}, W^V \in \mathbb{R}^{d_{model} \times d_v}$ . Self-attention od  $Q, K, V$  se definiše na sledeći način:

$$\text{Self-Attention}(Q, K, V, M) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

$QK^T$  je dimenzije  $T \times T$ , i ova matrica može da se interpretira kao matrica sličnosti između ulaznih tokena.  $M \in \mathbb{R}^{T \times T}$  je **matrica maskiranja**, ona nije od posebnog značaja za enkoder komponentu, već se koristi za dekodner, možemo uzeti u nastavku da je  $M = 0^{T \times T}$ . Pošto ove sličnosti nisu normalizovane (primijenjuje se obični skalarni proizvod) primijenjuje se softmax transformacija po vrstama od  $QK^T$  -  $(QK^T)_i$  predstavlja sličnosti tokena  $i$  sa svim ostalim tokenima, ove sličnosti prevodu se u kategoričku raspodjelu softmax transformacijom. Sličnosti se skaliraju sa  $\frac{1}{\sqrt{d_k}}$  kako bi se izbjegla saturacija softmax transformacije. Najzad ove sličnosti se koriste kako bi se svaki token izrazio kao linearna kombinacija svih ostalih tokena,  $o_i = \sum_{j=1}^T \alpha_{ij} V_j$ . Dakle, veće  $\alpha_{ij}$  povlači i veću kontribuciju tokena  $j$  u izlazu za token  $i$ . Izlaz ove operacije je dimenzije  $T \times d_v$ .



Slika 10: Vizuelizacija kompatibilnosti između tokena unutar self-attention operacije. Uočavamo da je za token **fox** kompatibilan sa tokenima **jumps**, **dog**. Izvor: link

Koeficijenti kompatibilnost se izračunavaju istovremeno i nezavisno, što znači da transformer familija obrađuje ulazni dokument "odjednom", a to nije bio slučaj kod rekurentnih neuronskih mreža. Vidimo i veliki potencijal za paralelizaciju, zbog kojeg su u tom smislu ove arhitekture veoma efikasne. Štaviše, transformer familija dodatno generalizuje ovo, tako što uvodi više paralelnih attention operacije, koje se paralelno izračunavaju. Autori [22] nazivaju ovu operacija **Multi-Head Attention**. Imamo:

$$\text{MHAttention}(H, M) = \text{concat}(\text{head}_1(H, M), \dots, \text{head}_h(H, M))W^O$$

$$\text{head}_i = \text{Self-Attention}(HW_i^Q, HW_i^K, HW_i^V, M)$$

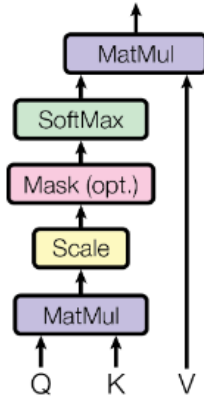
gdje su  $W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_v \in \mathbb{R}^{d_{\text{model}} \times d_v}, M \in \mathbb{R}^{T \times T}$  je matrica maskiranja,  $h$  je broj paralelnih "glava" koje izračunavaju svoje koeficijente kompatibilnosti, *concat* je operator konkatencije po kolonama, a  $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ . U praksi se gotovo uvijek uzima  $d_k = d_v = \frac{d_{\text{model}}}{h}$ , pa je izlaz dimenzionalnosti  $T \times d_{\text{model}}$  - iste kao i ulaz.

### 2.1.3 Rezidualne konekcije, normalizacija i MLP

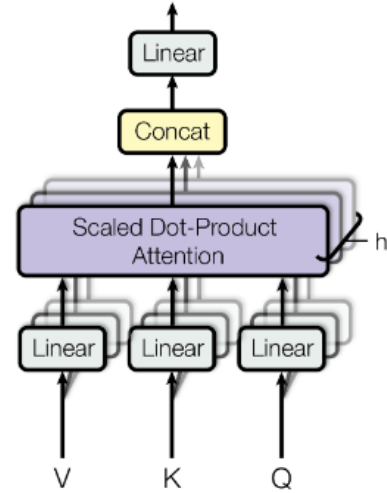
Na slici 9 uočavamo da se na više mjesta ulaz jedne operacije sabira sa izlazom druge operacije. Upravo ove veze na dijagramu 9 jesu **rezidualne konekcije**. Prvi put su izložene u članku [5], i uočeno je da je ova modifikacija itekako korisna prilikom treniranja vrlo dubokih neuronskih mreža. Kod ovakvih modela javlja se vanishing gradient problem, slično smo imali i za rekurentne neuronske mreže, i empirijski je uočeno da je ova jednostavna modifikacija vrlo korisna u borbi protiv ovog problema. Na konkretnom primjeru, imamo da je izlaz iz multi-head attention operacije zapravo  $H + \text{MH-Attention}(H, M)$ .

Detalj koji je izostavljen u članku [22], a koji su neki kasniji modeli "ispravili" jeste skaliranje ove konekcije sa  $\frac{1}{\sqrt{2}}$ . Ovo je motivisano

Scaled Dot-Product Attention



Multi-Head Attention



Slika 11: Vizuelizacija multi-head attention operacije. Izvor: [22]

sledećim: neka su  $X, Y$  dvije nezavisne slučajne veličine sa istom disperzijom, onda je:  $Var[X + Y] = Var[X] + Var[Y] = 2\sigma^2$ . Bez skaliranja sume od  $X + Y$  imali bi udvostručavanje disperzije, što nije povoljno za treniranje. U konkretnom primjeru, korekcija bi bila:  $\frac{H+MH-Attention(H,M)}{\sqrt{2}}$ .

Što se tiče normalizacije, primijenjuje se klasični **LayerNorm**. Ako je  $H_i \in \mathbb{R}^{d_{model}}$  ulazni vektor,  $E[H_i] = \sum_{j=1}^{d_{model}} h_{ij}$ ,  $Var[H_i] = \frac{1}{\sqrt{d_{model}-1}}(h_{ij} - E[H_i])^2$  onda imamo:

$$LayerNorm(H_i) = \frac{H_i - E[H_i]}{\sqrt{Var[H_i] + \epsilon}}\gamma + \beta$$

$\gamma, \beta \in \mathbb{R}^{d_{model}}$  se tretiraju kao naučivi parametri.

Ostalo je da objasnimo samo Feed Forward sloj. U pitanju je klasična **MLP - Multi Layer Perceptron** neuronska mreža, sa jednim skrivenim

slojem, i ona se može opisati u jednoj jednakosti:

$$FFN(x) = f(xW_1 + b_1)W_2 + b_2$$

gdje su  $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}, b_1 \in \mathbb{R}^{d_{ff}}, W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}, b_2 \in \mathbb{R}^{d_{model}}$ . Najčešće se uzima da je  $d_{ff}$  umnožak od  $d_{model}$ , autori [22] uzimaju  $d_{ff} = 4d_{model}$ .  $f$  je naravno funkcija aktivacije, autori [22] koriste **ReLU - Rectified Linear Unit**, neki moderniji modeli koriste **GeLU - Gaussian error Linear Unit**:

$$f(x) = x\Phi(x)$$

gdje je  $\Phi(x)$  funkcija raspodjele za  $\mathcal{N}(0, 1)$ . Ovim je opisan jedan **blok** transformer arhitekture. Enkoder (a u velikoj mjeri i dekodeer) je predstavljen sa  $N$  uzastopnih transformer blokova.

#### 2.1.4 Specifičnosti dekodeer komponente

Vraćajući se na diajgram 9, vidimo da su enkoder i dekodeer komponente vrlo slične, u ovoj sekciji ćemo se fokusirati na razlike:

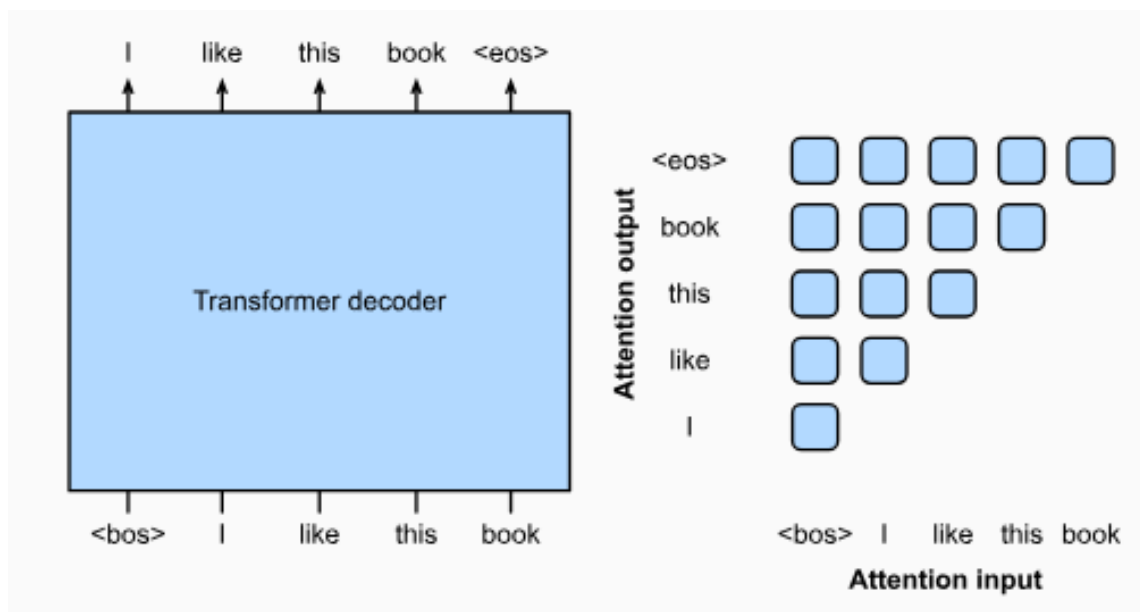
- Vektorska reprezentacija tokena  $E$  može, ali i ne mora da se dijeli između enkoder i dekodeer komponenti, zavisi od samog problema. Na primjer, ako koristimo enkoder-dekodeer arhitekturu za **mašinsko prevođenje** sa Engleskog na Kineski, pored razdvajanja ovih reprezentacija bilo bi smisleno razdvojiti i same algoritme za tokenizaciju, jer se ova dva jezika značajno razlikuju. U krajnjoj liniji, imali bi dva prolaza: konstrukcija vokabulara nekim algoritmom za tokenizaciju i inicijalizacija tabele tokena, za oba jezika. Sa druge strane, ako rješavamo problem **sumarizacije teksta**, originalni tekst i sumarizacija su na istom jeziku, pa nam je korisno da enkoder i dekodeer dijele isti vokabular, i iste reprezentacije tokena.
- U sekciji 2.1.2 pomenuli smo matricu maskiranja  $M$  prilikom primjene multi-head attention operacije (prvi multi-head attention sloj unutar jednog dekodeer bloka), i da je ona relevantna

samo za dekodera, sada ćemo objasniti i na koji način. Razmotrimo autoregressive language modelling problem, za ovu instancu dovoljan je samo dekodera, bez cross-attention operacije. Pretpostavimo da je jedan od ulaznih dokumenata  $d = \langle \text{start} \rangle \text{I like this book} \langle \text{end} \rangle$ . Tokom pohranjivanja u model, uzećemo  $d_1 = \langle \text{start} \rangle \text{I like this book}$ , a ciljani niz tokena će biti  $d_2 = \text{I like this book} \langle \text{end} \rangle$ . Znamo da će izlaz iz dekodera komponente nakon procesiranja  $d_1$  biti dimenzije  $H = T \times d_{\text{model}}$ , kako bi dobili logit koeficijente za vjerovatnoće treba nam  $|V|$  u drugoj dimenziji, to možemo postići uvođenjem nove linearne transformacije, ili češće  $HE_d^T$ , gdje su  $E_d \in \mathbb{R}^{|V| \times d_{\text{model}}}$  dekodera reprezentacije tokena. Problem nastaje u self-attention operaciji, jer sa  $QK^T$  mi računamo sličnosti između svaka dva para tokena, a ovo nije konzistentno sa autoregresivnom prirodom ovog problema. Konkretno u ovom primjeru, nakon tokena **this** sledeći najvjerovatniji token treba da bude **book**, ali u self-attention operaciji reprezentacija za **this** se računa kao linearna kombinacija svih tokena, a ne samo onih tokena koji prethode tokenu **this**. Zbog ovoga, prilikom računanja koeficijenata kompatibilnosti, treba da postavimo  $\alpha_{ij} = 0$  kad god je  $j > i$ . Ovo se lako postiže matricom maskiranja, imajući još u vidu da je  $e^{-\infty} = 0$ . Naime, dovoljno je postaviti:

$$M = \begin{bmatrix} 0 & -\infty & -\infty & \dots & -\infty \\ 0 & 0 & -\infty & \dots & -\infty \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{T \times T}$$

Drugim riječima, gornji trougao od  $QK^T$  popunjavamo sa  $-\infty$  kako bi ostvarili prethodno ograničenje. Dodatna ilustracija ovog primjera se može vidjeti na 12. Self-attention operacija sa primjenom ove matrice maskiranja naziva se i **causal self-attention**.

- Ovako dobijeni modeli - dekodera treniran za autoregressive language modelling nazivaju se još i **ULM - Unconditional Lan-**



Slika 12: Autoregressive language modelling kroz dekodeer komponentu transformera. Izvor: [link](#)

**guage Models**, jer koristimo samo dekodeer komponentu. Pojam unconditional može na prvo čitanje biti zbunjujući, pošto ne koristimo enkoder komponentu nemamo primjenu cross-attention operacije pa izlaz dekodeer komponente nije uslovljen izlazom enkoder komponente.

### 2.1.5 Specifičnosti enkoder-dekodeer modela

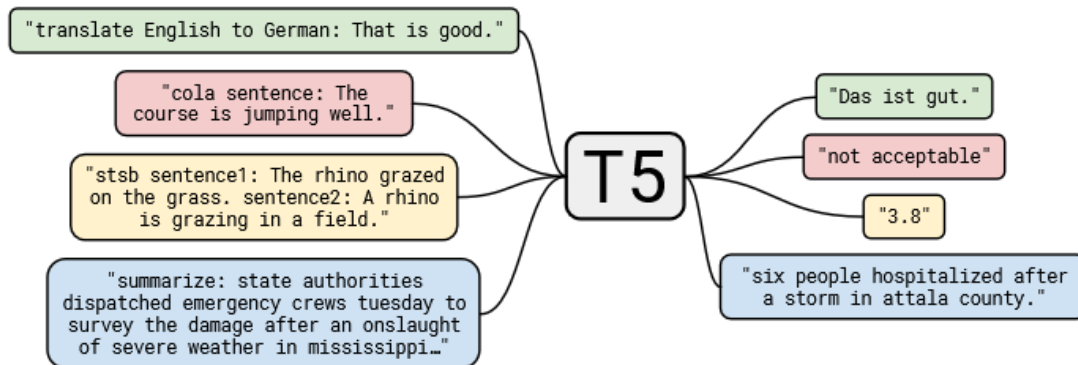
Ostaje samo da opišemo kako se enkoder nadovezuje na dekodeer u punoj enkoder-dekodeer arhitekturi. Imamo par rečenica, i kao što možemo vidjeti na slici 9 jedna ulazi u enkoder unutar kog se primjenjuje self-attention operacija, druga ulazi u dekodeer unutar kog na svakom nivou imamo dvije attention operacije - prva uzima samo izlaz prethodnog nivoa i primjenjuje causal self-attention, a druga uzima izlaz iz prethodne attention operacije i izlaz iz enkoder modela i primjenjuje **cross-attention** operaciju. Preciznije, cross-attention je suštinski isti kao self-attention, samo što se kombinuju reprezentacije enkoder i dekodeer modela.  $head_i$  za cross-attention računamo kao:

$$Q = HW_i^Q, K = ZW_i^K, V = ZW_i^V$$

gdje je  $Z$  izlaz iz enkodera i  $H$  izlaz iz prethodne operacije dekodera. cross-attention također koristi "praznu" matricu maskiranja,  $M = 0^{T \times T}$ . Na ovaj način je uspostavljena veza između enkodera i dekodera - izlaz dekodera je uslovljen izlazom enkodera. Ovakvi modeli se još nazivaju **CLM - Conditional Language models** - izlaz dekodera je sada uslovljen izlazom iz enkodera. Ako je cilj samo autoregressive language modelling u praksi se kao sasvim dovoljna pokazala decoder-only arhitektura, primjere ovakvih modela pomenuli smo na početku ovog poglavlja. Cross-attention operacije specifična je za enkoder-dekoder arhitekturu, ako imamo samo dekoder primijenjuje se samo causal self-attention!

Međutim, ispostavlja se i da enkoder-dekoder modeli mogu biti itekako korisni. U poglavlju u kom smo dali kratak pregled rekurentnih neuronskih mreža vidjeli smo da se problem mašinskog prevođenja suštinski svodi da učenje preslikavanja iz jedne rečenice u drugu, 4. Ova ideja može da se uopšti, gotovo svaki NLP problem možemo da posmatramo kroz **sequence-to-sequence framework** - mapiranje rečenice u drugu koja joj najviše odgovara. Na primjer, ako imamo dataset za klasifikaciju rečenica, očigledno je kako možemo da ovo pretvorimo u dataset pogodan za enkoder-dekoder modle. Neka je  $(X, y)$  jedna instanca za klasifikaciju iz polaznog dataset-a,  $X$  je rečenica a  $y$  je odgovarajuća labela. Ako su  $\{y_1, \dots, y_N\}$  sve klase koje se javljaju u ulaznom dataset-u, onda  $X' = \text{"Classify this sentence as one of } (y_1, \dots, y_N)\text{"}$ :  $X$ ,  $y' = \text{"Result"}$ :  $y$  može da se interpretira od strane enkoder-dekoder modela. Ovaj primjer bio je samo ilustrativan, postoje različiti algoritmi formatiranja  $(X, y)$ . Ilustraciju ovog koncepta možemo vidjeti na slici 13.





Slika 13: Kako se različiti NLP problemi mogu gledati kroz text-to-text (odnosno sequence-to-sequence) framework. Izvor: [14]

## 2.2 Enkoder modeli

Kao što smo ranije naveli, za problem klasifikacije teksta dovoljna je samo enkoder komponenta, tako da se u nastavku fokusiramo na neke od najpopularnijih transformer enkoder modela.

### 2.2.1 BERT

**BERT - Bidirectional Encoder Representations from Transformers**, prvi put objavljen u članku [4] od strane Google istraživača, sigurno je najpopularniji enkoder transformer model. Vrlo su sitne razlike u odnosu na klasičnu enkoder transformer komponentu:

- BERT koristi WordPiece [18] algoritam za tokenizaciju, nema nekih suštinskih razlika u poređenju sa prethodno izloženim BPE algoritmom. WordPiece vokabular je kardinalnosti  $\sim 3 \times 10^4$
- BERT uzima par rečenica (vidjećemo zbog čega u nastavku). Pored matrice za reprezentaciju tokena i pozicionih enkodiranja, BERT čuva tzv. **segment embeddings**, kako bi razdvojio koji token pripada kojoj rečenici. Ako je token iz prve rečenice onda se na njega dodaje  $a$ , a ako je iz druge rečenice onda se na njega dodaje  $b$ ,  $a, b \in \mathbb{R}^{d_{model}}$ . Ilustraciju ovog možemo vidjeti na 14.

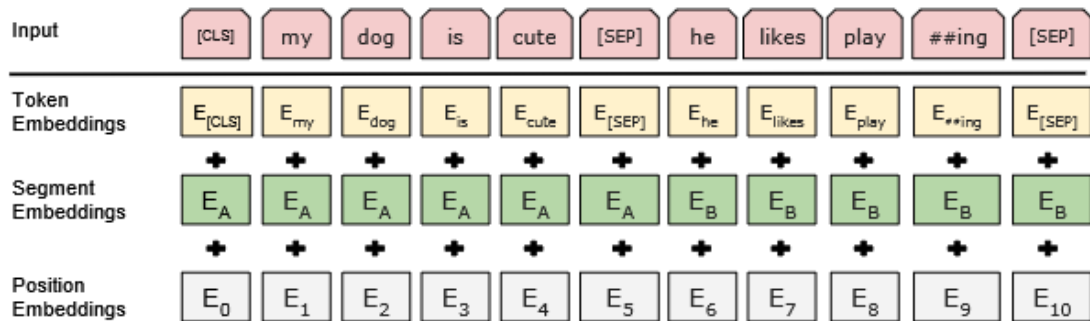
Dodatno, ulogu <start>tokena sada preuzima [CLS] token, a na kraju svake od ulaznih rečenica stavlja se specijalni token [SEP].

- Tokom **pre-training faze**, BERT se trenira nad sledećim problemima: **MLM - Masked Language Modelling** i **NSP - Next Sentence Prediction**. Kod MLM-a, za svaku ulaznu rečenicu se nad 15% ulaznih tokena primijenjuje sledeća pravila:
  - U 80% slučajeva se odabrani tokeni zamjenjuju sa novim, specijalnim tokenom [MASK].
  - U 10% slučajeva odabrani tokeni ostaju isti
  - Inače se odabrani tokeni zamjenjuju sa slučajno odabranim tokenima iz vokabulara.

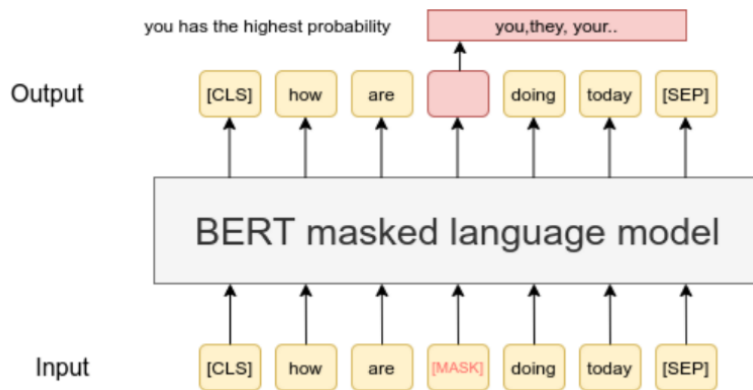
U bilo koje od prethodne 3 situacije, cilj je predvidjeti originalni token. Logit koeficijenti dobijaju se iz BERT reprezentacije [MASK] tokena. Na ovaj način model se "ohrabruje" da razumije jezik - kako bi predvidio originalni token mora na nekom nivou razumijeti zadatu rečenicu. Naravno, prethodne vjerovatnoće je moguće modifikovati po želji, ovo su vrijednosti koje su autori odabrali nakon sprovođenja eksperimenata. Ilustracija MLM možemo vidjeti na slici 15.

- Cilj NSP-a je da "ohrabri" model da uči odnose između parova rečenica. Prilikom treniranja, u 50% slučajeva par rečenica ( $A$ ,  $B$ ) je takav da je  $B$  rečenica koja slijedi nakon rečenice  $A$ , a u preostalim slučajevima je  $B$  nasumično odabrana rečenica iz korpusa. Za predikciju da li je  $B$  sledbenik od  $A$  koristi se reprezentacija [CLS] tokena, a logit koeficijenti za predikciju mogu se dobiti primjenom linearne transformacije  $W \in \mathbb{R}^{d_{model} \times 2}$  koja se uči tokom treninga. Ilustraciju ovog postupka možemo vidjeti na slici 16

**Pre-training** faza sprovedena je nad unijom dva skupa podataka: **BookCorpus** (200M tokena) i **English Wikipedia**(2.5B tokena). Autori su objavili dvije varijante BERT modela, prateći prethodno



Slika 14: Ilustracija segment embeddings koncepta. Izvor: [4]



Slika 15: Ilustracija BERT MLM koncepta, izvor: link

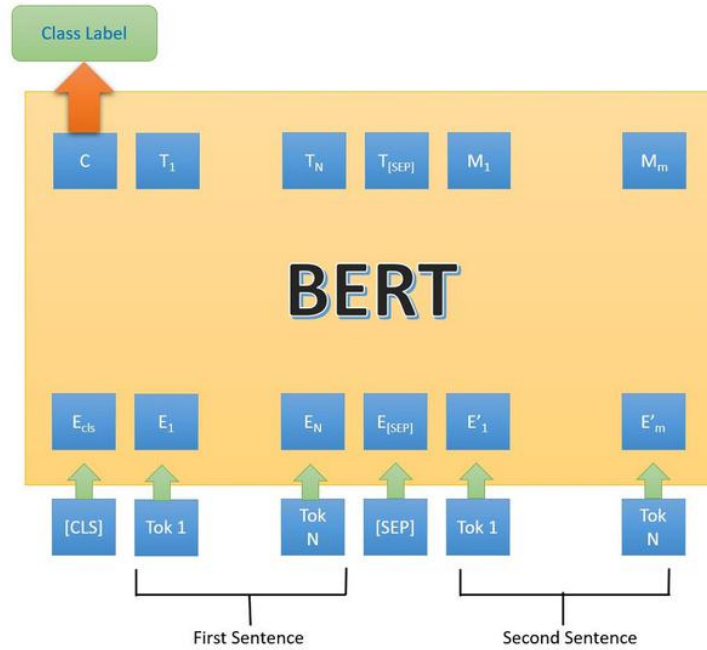
uvedenu terminologiju imamo:

$BERT_{BASE}(N = 12, d_{model} = 768, h = 12), \sim 110M$  parametara

$BERT_{LARGE}(N = 24, d_{model} = 1024, h = 16), \sim 340M$  parametara

Za rečenicu dužine  $T$  izlaz iz BERT modela je naravno dimenzionalnosti  $T \times d_{model}$ . Ovi vektori mogu da se razumiju kao kontekstualne reprezentacije tokena, engl. **contextual embeddings**. Kontekstualna upravo zbog self-attention operacije, jedan token će imati više različitih reprezentacija u zavisnosti od konteksta - preostalih tokena rečenice.

Dodatno, autori su pokazali da su rezultujuć modela vrlo pogodni

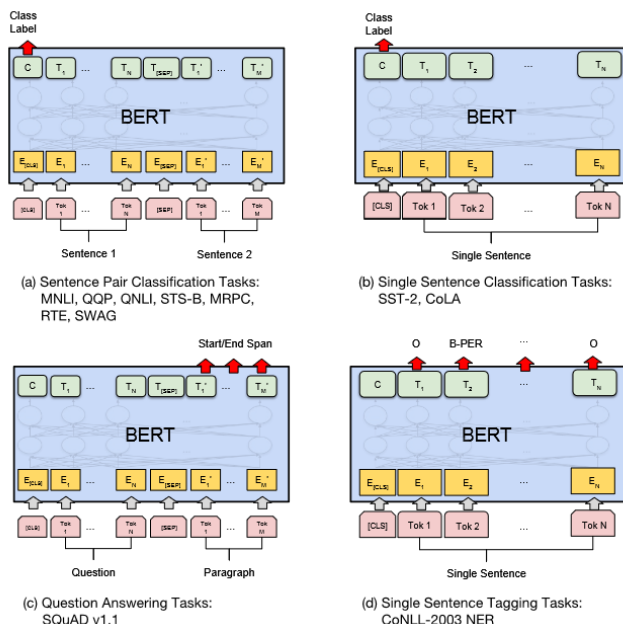


Slika 16: Ilustracija BERT NSP koncepta, izvor: [link](#)

za **transfer learning** paradigmu. Intuitivno, tokom pre-training faze model uči semantiku jezika nad podacima velikog obima, a zatim tokom **fine-tuning** faze koristi ovo znanje kako bi riješio specifičniji problem kao što je klasifikacija, zahtijevajući podatke mnogo manjeg obima. Konkretno za klasifikaciju teksta/rečenica, tokom fine-tuning faze, u BERT pohranjujemo par rečenica  $(A, \emptyset)$  (od interesa je klasifikacija samo jedne rečenice, ne parova rečenica). Strategija koja se najčešće primijenjuje jeste da se uzme kontekstualna reprezentacija [CLS] tokena, a zatim ako je  $K$  broj klasa koje treba odrediti, logit koeficijente računamo kao  $WH_{[CLS]}$ , gdje je  $W \in \mathbb{R}^{K \times d_{model}}$  transformacija koja se uči tokom fine-tuning procedure. Jasno, ovaj model koji se "kači" na reprezentaciju [CLS] može biti proizvoljne složenosti. Dodatno, tokom fine-tuning moguće je ograničiti broj parametara koji se trenira, prije svega zbog hardverskih nedostataka konvencionalnih računara, ali i ovome će biti više riječi u nekoj od narednih sekcija.

Na slici 17 možemo vidjeti kako uz minimalne modifikacije možemo

vršiti fine-tuning BERT modela za različite probleme:



Slika 17: Fine-tuning BERT modela. Varijanta c) je za nijansu komplikovanija od preostalih, za nejasnoće konsultovati referencirani članak. Izvor: [4]

## 2.2.2 RoBERTa

**RoBERTa - Robustly optimized BERT approach.** Sitne promjene u BERT pre-training metodama koje su dovele do state-of-the-art rezultate, u vremenu objavljivanja članka [11]. U pitanju su sledeće modifikacije:

- Iz prethodne formulacije mask language modelling-a zaključilo bi se da prilikom pohrane rečenice u model tokom pre-training faze, biramo nasumično 15% tokena koji će biti maskirani, tako je ovaj proces opisan i u samom BERT članku. Međutim, istražujući originalni BERT repozitorijum, autori RoBERTa su uočili da se odabir tokena za maskiranje dešava samo jednom prilikom pre-procesiranja podataka, što značajno smanjuje teori-

jski broj rečenica koje mogu biti prikazane modelu tokom pre-training procedure, i očekivano, negativno utiče na razumijevanje jezika. Autori su ispravili ovaj "bug", i maskiranje kod RoBERTa je konzistentno sa originalnom definicijom.

- Povećavanje batch size parametra - broj parova rečenica koji se paralelno pohranjuju u model prilikom pre-training faze, ovaj parametar relevantan je i za fine-tuning fazu.
- Drugačija šema za preprocesiranje tokena. RoBERTa koristi sledeće:  $[S] \ x_1 \dots x_n \ [/S] \ y_1 \dots y_m \ [/S]$ .  $[S]$  sada preuzima ulogu  $[CLS]$  tokena, dok  $[/S]$  označava kraj ulazne rečenice.
- Uklanjanje segment embeddings objekta.
- Autori su uočili da uklanjanje NSP iz pre-training faze poboljšava rezultate, tako da je uklonjen. Naravno, RoBERTa dopušta NSP tokom fine-tuning faze, samo je uklonjen tokom pre-training faze.
- Korišćenje BPE algoritma umesto WordPiece, vokabular od  $\approx 5 \times 10^4$  tokena.
- Povećavanje obima pre-training podataka - rezultujući skup podataka je veličine  $\sim 160GB$ . Ovo povlači i da je vrijeme potrebno za **pre-training** duže: za BERT<sub>LARGE</sub> potrebno je nekih 6100 GPU sati, dok je za RoBERTa potrebno nekih 24000 GPU sati. U pogledu same arhitekture, RoBERTa je identična BERT<sub>LARGE</sub> modelu, dakle isti broj parametara, samo je obim pre-training podataka znatno veći.

Što se tiče samog fine-tuning-a, postupak za RoBERTa je identičan kao za BERT, tako da mu nećemo posebno posvećivati pažnju. U praksi je generalno uvijek realno za očekivati da će performanse RoBERTa modela biti bolje od običnog BERT modela.

### 2.2.3 Sentence BERT

Iako nisu semantički bogate (u smislu kosinusne sličnosti) kao word2vec reprezentacije, BERT je vrlo lako moguće obučiti da proizvodi semantički bogate reprezentacije. Jedan pristup je **Sentence BERT** - [15]. Već smo vidjeli da je izlaz iz BERT modela  $H \in \mathbb{R}^{T \times d_{model}}$ , za dokument dužine  $T$ , gdje je  $H_1$  kontekstualna reprezentacija [CLS] tokena. U principu, za jedan dokument moguće je generisati jedan vektor koji definiše njegovu BERT reprezentaciju, ovo su **operacije združivanja - pooling**. Tri potencijalne strategije su:

- Uzmianje  $H_1$ , to smo već imali kod prethodna dva modela.
- $\frac{1}{T} \sum_{i=1}^T H_i$ , računanje reprezentacije dokumenta kao prosječni vektor, ovo je podrazumijevani pristup Sentence BERT metoda.
- $\hat{H}_j = \max(H_{1j}, \dots, H_{Tj})$ , računanje maksimalne koordinate po svakoj dimenziji izlaznih vektora i združivanje u jedan vektor.

Pitanje je kako iskoristiti združenu reprezentaciju dokumenta da postignemo semantičko obogaćivanje (opet, u smislu kosinusne sličnosti). Autori Sentence BERT metoda izlažu 3 načina, mi ovdje izlažemo samo jedan koji je najpopularniji, preostala dva su veoma slična - za više detalja pogledati referencirani članak. Autori uzimaju kombinaciju **SNLI - Stanford Natural Language Inference** i **Multi Genre NLI** skupova podataka. Oba skupa sadrže parove rečenica  $(A, B)$  i labelu koja ima 3 moguće vrijednosti:

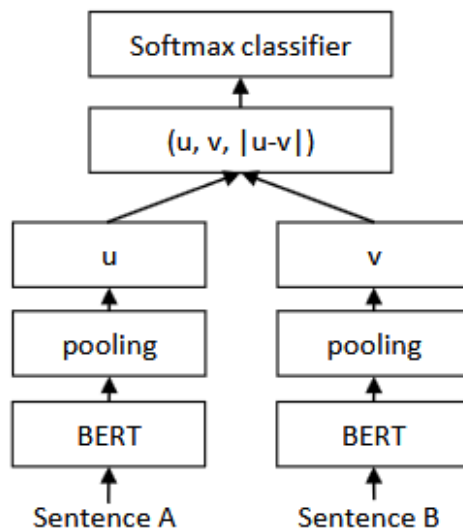
- **entailment - povlačenje**, rečenica  $B$  se može zaključiti iz rečenice  $A$ .
- **contradiction - kontradikcija**, rečenica  $B$  nije konzistentna sa rečenicom  $A$
- **neutral** - Ne može se zaključiti ništa

Postavka ovog problema je vrlo slična NSP problemu koji smo imali tokom pre-training faze BERT modela, samo sada imamo kategoričku promjenjivu sa 3 moguće vrijednosti, a prethodno su bile 2 moguće

vrijednosti. Neka su  $u, v$  združene BERT reprezentacije rečenica  $A, B$  redom. Semantičko obogaćivanje  $u, v$  postiže se tako što se modelira prethodno navedena kategorička promjenjiva:

$$p = \text{softmax}(W_t \text{concat}(u, v, |u - v|))$$

Gdje je  $W_t \in \mathbb{R}^{k \times 3d_{\text{model}}}$ , a  $k$  je broj mogućih vrijednosti za kategoričku veličinu koja se modelira, u ovom slučaju  $k = 3$ . Ilustraciju ovog postupka možemo vidjeti na slici 18



Slika 18: SBERT semantičko obogaćivanje kroz klasifikaciju parova rečenica. Izvor: [15]

Dakle, uzima se pre-trenirani BERT ili RoBERTa model, a onda se vrši prethodno opisani fine-tuning. Kada je u pitanju klasifikacija, prednost SBERT modela jeste što u praksi on ne zahtijeva dodatnu kalibraciju parametara - može da se koristi fine-tuned checkpoint iz prethodnog članka koji je javno dostupan, i koji proizvodi semantički bogate reprezentacije. Još jedna prednost ovog sistema je mogućnost za **zero-shot classification**. Naime, za prethodno opisane modele, da bi vršili fine-tuning nad skupom podataka za klasifikaciju (parova) rečenica, morali smo unaprijed da znamo broj željenih klasa, dok se



ovdje oslanjamo isključivo na kosinusnu sličnost i možemo je računati između proizvoljnih parova rečenica. Jedna mana je što iako pruža semantički bogate reprezentacije, ovaj model nije usko specijalizovan za klasifikaciju nad zadatim labelama, kao što bi bio fine-tuned BERT ili RoBERTa, pa je u praksi realno za očekivati da daje nešto lošije performanse.

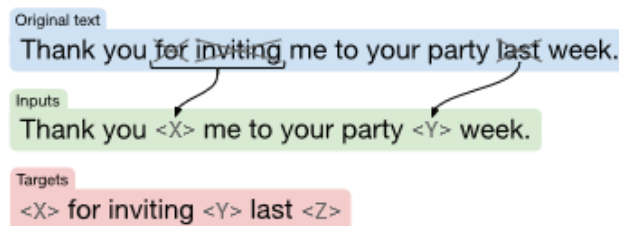
Na kraju, navodimo da se SBERT i slični sistemi danas vrlo često sreću u modernim information retrieval i search engine sistemima, a posebno ističemo primjenu ovakvih modela u **RAG - Retrieval Augmented Generation** metodama za decoder-based chat-botove. Jedan članak na ovu temu, vrlo sličan sa SBERT je [8].

## 2.3 Enkoder-dekoder modeli

### 2.3.1 T5

**Text to Text Transfer Transformer - T5** je familija transformer enkoder-dekoder modela koja pokušava da u potpunosti iskoristi sequence-to-sequence framework, [14]. Glavna motivacija bila je da se konstruiše jedan univerzalni pre-trained enkoder-dekoder model koji kroz sequence-to-sequence framework uz relativno malo vrijeme treniranja nakon pre-training faze (fine-tuning) može da se uopšti na sve NLP probleme. U jednoj od prethodnih sekcija opisali smo kako funkcionišu enkoder-dekoder modeli, po tom pitanju T5 se ne razlikuje od prethodno izloženog. Ono što nije očigledno je kako se uopšte može vršiti pre-training enkoder-dekoder modela. Sjetimo se da su trening instance za sequence-to-sequence framework parovi rečenica, a tokom pre-training faze tipično se skuplja masivna količina podataka sa Interneta i drugih izvora, i velika većina tih podataka nije označena, u smislu da se zna koja rečenica je prva a koja druga. Ovolika količina podataka zahtijeva da je algoritam treniranja **nenadgledan** (engl. **unsupervised**). Autori [14] uvode **span corruption loss**, koji je vrlo sličan BERT-ovom masked language modelling pristupu, i opisujemo ga u nastavku.

Razmatramo jednu ulaznu rečenicu i želimo da od date rečenice do-



Slika 19: Ilustracija konstrukcije trening instanci za T5 span corruption loss. Izvor: [14]

bijemo ulaz pogodan za sequence-to-sequence framework. Uvode se sledeći parametri:

- $p$  - vjerovatnoća da će ulazni token biti maskiran. Autori su se opredijelili za  $p = 0.15$
- $l$  - prosječna dužina podniza uzastopno maskiranih tokena. Autori su se opredijelili za  $l = 3$

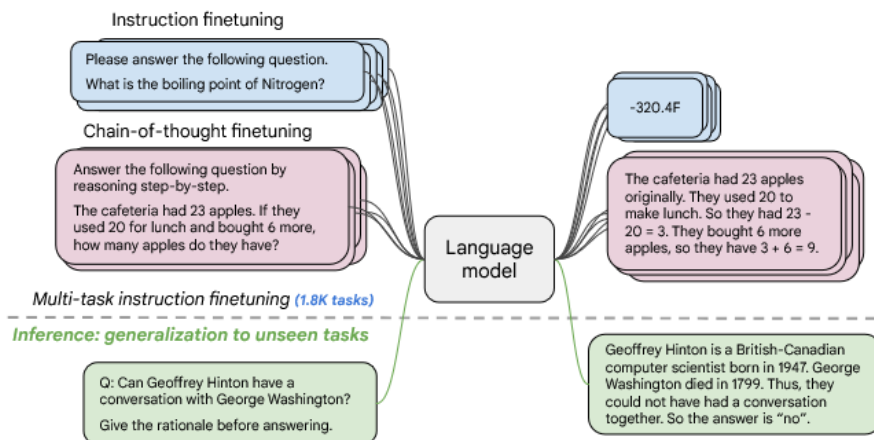
Na osnovu ova dva parametra primijenjuje se odgovarajuća transformacija nad ulaznom rečenicom, pri čemu je očekivani broj maskiranih podnizova tokena  $\lfloor \frac{x \times p}{l} \rfloor$ , gdje je  $x$  dužina ulazne rečenice. Svi maskirani podnizovi tokena zamjenjuju se specijalnim **sentinel** tokenima - ovi tokeni dodaju se nakon konstrukcije vokabulara i nezavisni su u odnosu na algoritam za konstrukciju vokabulara - ovakav je i [MASK] BERT token. Ulaz za enkoder će biti maskirana rečenica sa sentinel tokenima, a ciljanu rečenicu dobijamo tako što izvršimo "komplement" ulaza za enkoder - sve maskirane/sentinel tokene zamjenjujemo sa originalnim podnizovima tokena, a sve ne-maskirane tokene zamjenjujemo sa sentinel tokenima. Na ovaj način podstičemo model da "razumije" ulazni jezik, i omogućava nam generalizaciju na proizvoljne NLP probleme nakon pre-training faze. Ilustracija kako se konstruišu instance za span corruption loss može se vidjeti na slici 19. Sami dataset koji su koristili autori T5 je **C4 - Colossal Clean Crawled Corpus**, koji je veličine  $\approx 750GB$ .

U pogledu same arhitekture, T5 ne odstupa značajno od originalne

enkoder/dekoder arhitekture koju smo već uveli. Najveća razlika jeste drugačija šema za poziciono enkodiranje - T5 modeli koriste **relative positional encoding** šemu, a mi smo prethodno opisali **absolute positional encoding** šemu. Ipak, smatramo da ova šema nije suštinska za razumijevanje ovog modela, više matematičkih detalja o njoj mogu se naći u članku koji je uveo ovu šemu, [17]. Postoji više instanci iz T5 familije, osnovne su:

- T5<sub>SMALL</sub> -  $\approx 220M$  parametara
- T5<sub>BASE</sub> -  $\approx 60M$  parametara
- T5<sub>LARGE</sub> -  $\approx 770M$  parametara

Ako uzmemo u obzir broj parametara za BERT<sub>BASE</sub>  $\approx 110M$  i činjenicu da je ovo enkoder model, kao i da je T5 enkoder-dekoder model, onda je sasvim prirodno da je broj parametara za T5<sub>BASE</sub> približno duplo veći. Na kraju, navodimo jednu za praksu interesantnu varijantu T5 modela koju ćemo koristiti za ovaj problem - **FLAN T5**, [2]. Ovaj model kreće od T5 1.1 instance pre-trenirane za span corruption loss, i sprovodi opsežan sequence-to-sequence fine-tuning nad približno 1.8 hiljada dataset-ova visokog kvaliteta (u poređenju sa podacima za pre-training) kao što su SST-2 i SQuAD kako bi proizveli general-purpose model, ilustracija ovog postupka može se vidjeti na slici 20. Primijetimo da među ovim dataset-ovima imamo različite kategorije NLP problema, SST-2 je dataset za klasifikaciju teksta dok je SQuAD dataset za question answering. Međutim, pošto koristimo sequence-to-sequence paradigmu ovo nam ne predstavlja nikakav problem, samo je potrebno primijeniti odgovarajuće transformacije nad ulaznim rečenicama. Napominjemo da postoje određene razlike između T5 1.1 i originalnog T5 modela, ali nisu toliko značajne pa ih nećemo navoditi ovdje. Za više informacija pogledati link (za T5 1.1 nije objavljen poseban članak).



Slika 20: Pregled instruction-based fine-tuning procedure za FLAN-T5 na visokom nivou. Primijetimo da se terminologija na ovoj slici ne poklapa sa našim konvencijama - ovdje se enkoder/dekoder model oslovljava sa **language model**, za nas je to **conditional language model**!

### 3 Praktična razmatranja

Za ovaj projekat, koristimo ovaj skup podataka, dostupan na Kaggle platformi. Plan je da se uporede sledeći pristupi:

- fine-tuning RoBERTa instance.
- zero-shot classification koristeći SBERT, bez fine-tuning procedure.
- fine-tuning FLAN-T5 instance.

Prije upoređivanja ovih modela, moramo ispitati svojstva zadatog dataset-a, tako da se sa tim u vezi prvo fokusiramo na **EDA - Exploratory data analysis** i preprocesiranje podataka. Diskutovaćemo i o odabiru metrike za poređenje ovih modela, kao i o dodatnim praktičnim razmatranjima koja se tiču fine-tuning velikih transformer modela u praksi.

### 3.1 Podaci, augmentacija i preprocesiranje

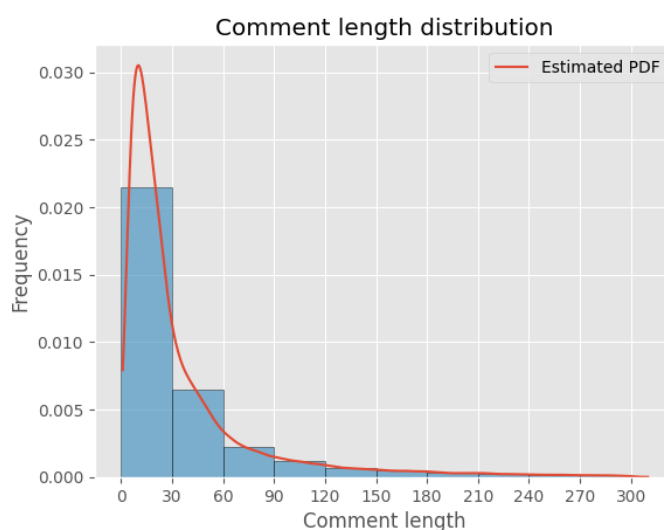
U Kaggle repozitorijumu mogu se naći dva dataset-a, **HateSpeechDataset.csv** i **HateSpeechDatasetBalanced.csv**. Prvi broji  $\approx 410000$  komentara, od kojih  $\approx 86\%$  nisu klasifikovani kao govor mržnje. Uočavamo neravnomjernu raspodjelu klasa, autori su ovaj problem riješili na sledeći način:

- Rijeđe uzorkovanje rečenica koje ne predstavljaju govor mržnje prilikom konstrukcije novog skupa podataka, **undersampling**
- **Augmentacija** rečenica koje predstavljaju govor mržnje. Konkretno, od jedne rečenice  $d$  konstruišu više rečenica supstitucijom postojećih riječi/dodavanjem sinonima.
  - **Supstitucija** postojećih riječi vršena je koristeći osnovni BERT model, jasno je da ako slučajno odabrani token zadata rečenice zamijenimo sa [MASK] tokenom, na izlazu iz BERT modela moći ćemo da konstruišemo raspodjelu najvjerovatnijih originalnih tokena. Nova rečenica upravo se dobija uzorkovanjem nad ovom raspodjelom. Ovdje vidimo još jedan način korišćenja BERT-like modela u praksi!
  - **Dodavanje sinonima** vršeno je koristeći **WordNet**, [21], bazu podataka koja za zadatu riječ, između ostalog, može da odgovara na upite oblika "koji su sinonimi zadate riječi". Značajno se razlikuje od prethodno opisanih sistema pa nećemo ulaziti u detalje, više informacija može se naći u prethodnoj referenci. Jasno je da bismo mogli primijeniti i prethodno opisane word2vec i SBERT sisteme za augmentaciju podataka.

Rezultujući skup podataka ima  $\approx 700000$  rečenica, a raspodjela labela je približno jednaka, što je mnogo prikladnije za trening, u smislu ravnomjerne raspodjele klasa, tako da ćemo za trening modela koristiti upravo ovaj dataset, dakle **HateSpeechDatasetBalanced.csv**. Preciznije, odrađena je 80/20 podjela na podatke za trening i podatke

za testiranje, sa tim da su ovo konfigurabilni parametri.

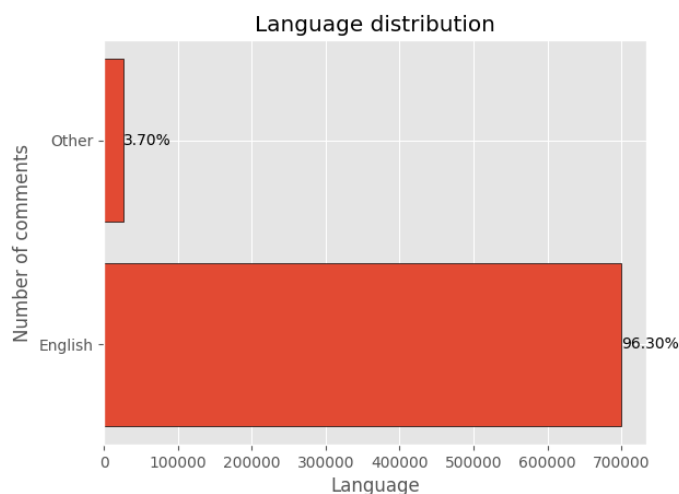
Originalni dataset, **HateSpeechDataset.csv** sadržao je cased rečenice (velika i mala slova prisutna), dok transformisani dataset sadrži lowercased rečenice (sve rečenice zapisane malim slovima). Intuitivno, očekujemo da bi casing mogao da bude važan za klasifikaciju u ovom domenu, na primjer za rečenice "I hate you" i "I HATE YOU", čovjek bi potencijalno dao veću težinu drugoj rečenici nego prvoj. Međutim, lowercasing nije reverzibilna transformacija, tako da korišćenjem balansirane varijante ovog dataset-a trajno gubimo originalni oblik rečenice, pa nismo testirali ovu hipotezu. Nad balansiranom varijantom vršene su dodatne klasične operacije preprocesiranja, kao što je uklanjanje ponovljenih bjelina, pa nam ovo omogućava da ispitamo raspodjelu broja riječi po rečenicama, to možemo vidjeti na slici 21.



Slika 21: Raspodjela broja riječi u rečenicama balansiranog dataset-a

Dodatno, ručnom validacijom rečenica uočeno je da balansirana varijanta dataset-a sadrži više jezika. Kako bi barem grubo utvrdili koji su jezici prisutni, koristili smo langid klasifikator, koji nije savršen ali nam pruža dovoljno dobru aproksimaciju raspodjele jezika koju možemo vidjeti na slici 22. Ispostavlja se da je  $\approx 96.3\%$  rečenica na engleskom jeziku, dok svi ostali prisutni jezici čine  $\approx 3.7\%$  ukupnog

broja rečenica. Pri kreiranju grafika 22 uveli smo heuristiku da je jezik značajan ako ima barem 10% rečenica/komentara (takođe konfigurabilno), a inače se svrstava u "Other" kategoriju. U situacijama kada raspodjela jezika nije homogena kao ovdje, jedna od opcija bi bila da se koristi **cross-lingual** (u nekoj literaturi i **multi-lingual**) model, ovi modeli su kratko opisani u jednoj od narednih sekcija.



Slika 22: Raspodjela jezika u balansiranom dataset-u, koristeći langid klasifikator jezika

Na kraju, napominjemo da kada je u pitanju fine-tuning tekstualnih transformer modela, najčešće se ne primjenjuje nikavko preprocesiranje osim eventualne augmentacije podataka. U nekom smislu, sami algoritam za tokenizaciju koji smo odabrali vrši preprocesiranje teksta, i nema potrebe za komplikovanim jezičkim transformacijama kao što je lematizacija koju smo prije pomenuli.

## 3.2 LoRA - Low Rank Adaptation

U BERT sekciji, izložili smo dvije varijante BERT modela koje se razlikuju samo u broju parametara, BERT<sub>BASE</sub> i BERT<sub>LARGE</sub>, koje broje  $\approx 110M$ ,  $\approx 340M$  parametara redom, iste kardinalnosti su RoBERTa varijante. Izložili smo i broj potrebnih GPU sati za pre-training fazu:

6100 GPU sati za BERT<sub>LARGE</sub>, 24000 GPU sati za RoBERTa<sub>LARGE</sub>. Ovi brojevi daleko nadmašuju kapacitete konvencionalnih računara. Iako je fine-tuning suštinski isto što i pre-training faza, samo nad podacima znatno manjeg obima, hardverska složenosti pre-training faze se jednim dijelom prenosi i na fine-tuning fazu. Za našu primjenu, na visokom nivou, imamo neuronsku mrežu od  $\approx 340M$  parametara koja rješava problem klasifikacije, što znači da se trening svodi na primjenu **backpropagation** algoritma nad **cross-entropy** funkcijom. Ako uzmemo u obzir broj parametara, jasno je da je ovo vrlo složena operacija, i vremenski i memorijski. Dodatno, svi ovi problemi su duplo izraženiji kod enkoder-dekoder modela kao što je T5.

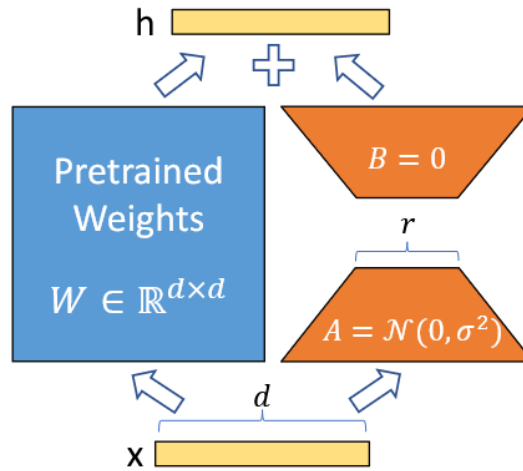
Jedna mogućnost je da se ograniči broj parametara originalnog modela koji se trenira (engl. **trainable parameters**). Na primjeru BERT modela, u praksi se najčešće realizuje tako što se trenira linearna transformacija za logit koeficijente klasifikacije (engl. **classification head**), i samo nekoliko poslednjih slojeva ovog modela. Ovo uvodi dodatni hiperparametar - broj slojeva koji se treniraju, što usložnjava proces treniranja u praksi. Alternativa koja je u poslednje vrijeme vrlo popularna jeste **LoRA - Low rank Adaptation**. Ovaj mehanizam prvi put je opisan u [6], od strane Microsoft istraživača. Neka je  $W \in \mathbb{R}^{m \times n}$  linearna transformacija u sklopu postojećeg modela. Na primjeru BERT modela, imamo linearne transformacije unutar MLP slojeva i attention slojeva, kao i linearnu transformaciju za logit koeficijente za klasifikaciju, ali se poslednja često izostavlja iz LoRA algoritma jer nije memorijski zahtjevnija kao prethodne. Uvodimo nove matrice  $A, \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}$ , gdje je  $r \in \mathbb{N}$  hiperparametar koji se naziva **rang**,  $r \ll \min(m, n)$ . Definišemo novu linearnu transformaciju:

$$\Delta W = W + \alpha AB$$

Gdje god se javlja transformacija  $W$ , mijenjamo je sa  $\Delta W$ . Tokom optimizacije, mijenjamo samo  $A, B$ ,  $W$  ostaje ista i ne figuriše tokom izvršavanja backpropagation algoritma. Razmotrimo prostornu uštedu, prethodno smo kroz backpropagation optimizovali po  $W$  sa  $mn$  vri-



jednosti, a sada optimizujemo po dvije matrice sa ukupno  $mr + rn$  vrijednosti. Imamo  $\frac{mr+rn}{mn} = \frac{r(m+n)}{mn}$ , a ako uzmemo u obzir da je  $r \ll \min(m, n)$ , uočavamo značajnu memorijsku efikasnost ovog metoda.  $\alpha > 0$  je hiperparametar koji određuje koliko nova LoRA linearna transformacija utiče na originalnu linearnu transformaciju. Nakon završetka treninga, vremenska složenost procesiranja dokumenata je ista kao i kod originalnog modela,  $W$  zamijenimo sa  $\Delta W$ , a pošto je trening završen  $A, B$  se neće mijenjati, što je izuzetno povoljno, i ne važi za sve **PeFT - Parameter efficient Fine Tuning** algoritme. Ilustraciju ovog algoritma možemo vidjeti na [6]. Prethodni metod može da se primjeni i na enkoder-dekoder modele bez ikakvih modifikacija - samo mijenjamo linearne transformacije unutar samog modela. Gotovo svi PEFT algoritmi implementirani su u sklopu peft Python biblioteke.



Slika 23: Ilustracija LoRA algoritma. Izvor: [6]

U praksi se pokazalo da je ovo mnogo ekonomičniji način treniranja od klasičnog fine-tuninga, prvenstveno zbog kalibracije hiperparametara (engl. **hyperparameter tuning**). Trening LoRA modela traje znatno kraće od treniranja originalnog modela, pa je samim tim moguće isprobati više kombinacija hiperparametara. Na kraju, napominjemo da iako se brojevi od  $\approx 340M$  parametara čine ogromnim, nije rijetkost

da se u praksi sreću još veći model. Primjer jednog ovakvog modela je Llama-3-8B, koji ima približno 8 milijardi parametara, a postoji i verzija od približno 70 milijardi parametara.

### 3.3 Izbor metrike za evaluaciju

Ako primijenimo SBERT, dobijamo semantički bogate reprezentacije, i možemo da mjerimo samo kosinusnu sličnost. Ovaj model nije probabilistički, i stoga nije direktno uporediv sa RoBERTa modelom. Iako bi SBERT bez većih komplikacija mogao da se prevede u probabilistički model, T5 se ne može ni na koji način prevesti u probabilistički model, jer u sequence-to-sequence paradigmi za ulaznu rečenicu na izlazu dobijamo novu rečenicu, a ne vjerovatnosnu raspodjelu. Štaviše, kada se završi trening modela moguće je da za istu rečenicu na uzalu dobijemo više različitih rečenica na izlazu, u zavisnosti od toga koji algoritam za autoregresivno dekodiranje smo odabrali (najčešće je to **top-p sampling + temperature softmax**). Za evaluaciju probabilističkih modela vrlo često se koristi macro/weighted **averaged ROC-AUC**. Ne samo da kvantifikuje performanse samog modela, već na osnovu ove metrike možemo i da vidimo kako promjena praga pouzdanosti utiče na **false positive rate**, što može biti izuzetno korisno u praksi. Međutim, ne možemo koristiti ROC-AUC nad modelom koji nije probabilistički kao što su T5 i izvorni SBERT, pa stoga predlažemo sledeći pristup.

Za fiksirani prag pouzdanosti RoBERT-a modela, lako računamo  $F_1$  metriku. Neka su  $c_1, \dots, c_n$  klase od interesa, pri čemu svakoj klasi pridružujemo tekstualnu labelu  $t_1, \dots, t_n$ . Konkretno u našem primeru imali bi dvije labele "This text is hate-speech", "This text is not hate-speech". Naravno, na performanse SBERT modela utiču i labele koje smo odabrali, i mogu da se tretiraju kao hiperparametri za zero-shot classification. Za ulazni dokument  $d$  i SBERT model računamo kosinusne sličnosti između  $d$  i zadatih labela, i uzimamo klasu čija je tekstualna labela najbližnja sa  $d$ :

$$x = \operatorname{argmax}_i E(d)E(t_i)^T$$

pri čemu pretpostavljamo da su vektori na izlazu iz  $E$  normirani. Iako i dalje ne raspolazemo sa vjerovatnosnom raspodjelom, direktno smo klasifikovali dokument  $d$ , pa možemo mjeriti **precision** i **recall** našeg modela, odnosno  $F_1$  kao uopštenje od ove dvije metrike. Što se tiče T5 modela, situacija je mnogo lakša, za ulaznu rečenicu model nam vraće pripadnu klasu u obliku tekstualne labele, ako se ova labele poklapa sa stvarnom labelom onda je tačna predikcija a inače nije. Prema tome, za evaluaciju sva 3 modela koristićemo  $F_1$  metriku.

### 3.4 Cross-lingual enkoderi

Pre-training svih prethodno opisanih modela je sproveden nad podacima na engleskom jeziku, što povlači da su ti modeli upotrebljivi samo za te jezike. Postavlja se pitanje kako ih primijeniti na podatke koji su na drugim jezicima. U ovoj sekciji izlažemo dvije metode:

- **XLM-R - Cross Lingual Language Model with RoBERTa**  
- Prvi put izložen u [3], predstavlja RoBERTa model za koji je pre-training vršen nad ogromnim skupom podataka koji sadrži 100 različitih jezika. **Cross-lingual** komponenta ovog modela ogleda se u tome što svi jezici dijele isti vokabular, koji broji nekih 250 hiljada tokena, značajno više u odnosu na BERT i RoBERTa. Na isti način je moguće je vršiti fine-tuning ovog modela polazeći od pre-trained varijante za bilo koji od jezika koji je podržan.
- XLM-R model enkodira 100 različitih jezika, i intuitivno očekujemo da nije usko specijalizovan ni za jedan, iako može sve da ih razumije. Očekujemo da će model koji je specijalizovan za uži skup jezika imati bolje performanse od XLM-R. Jedan članak koji je ovo potvrdio je [20]. Autori su uveli **CroSloEngual-BERT** model, za koji je pre-training sproveden nad podacima na 3 jezika: engleski, slovenački i hrvatski. Engleski je odabran

iz razloga što je skup podataka na ovom jeziku izuzetno bogat, znatno bogatiji u odnosu na hrvatski i slovenački. I ovaj model ima cross-lingual svojstvo, sva 3 jezika dijele isti vokabular. Autori su pokazali da ovako treniran model daje bolje performanse za svaki od prethodna 3 jezika, u odnosu na **mBERT - multilingual BERT** nakon sprovođenja fine-tuning procedura za probleme kao što su **NER - Named Entity Recognition**. mBERT je malo stariji model u odnosu na XLM-R, i on podržava 104 jezika, [10].

Suštinski, i XLM-R i CroSloEngualBERT, su instance BERT/RoBERTa modela, razlikuju skupovi podataka nad kojima je sproveden pre-training - višezječni, i svi jezici dijele zajedničku tabelu reprezentacija tokena. CroSloEngualBERT može biti posebno interesantan za rješavanje problema detekcije govora mržnje na crnogorskom jeziku, imajući u vidu sličnost između crnogorskog i hrvatskog jezika.

### 3.5 Knowledge distillation

Pored **quantization** tehnike, redukovanje memorijske složenosti koristeći reprezentacije realnih brojeva niže preciznosti, još jedna tehnika koja se često primjenjuje u praksi je **knowledge distillation**. Pretpostavimo da za problem klasifikacije imamo model  $T$ , i želimo na osnovu njega da kreiramo model  $S$  koji ima manje parametara od  $T$  sa prihvatljivim performansama. Razmatramo kategoričke raspodjele  $p_t, p_s$  koje su opisane modelima  $T, S$  redom. Ako su performanse od  $T$ , onda nam je cilj da  $p_s$  što više približimo  $p_t$ . Kako bi kvantifikovali koliko se raspodjela  $p_s$  razlikuje od raspodjele  $p_t$  koju uzimamo kao **ground-truth** možemo koristiti **Kullback-Leibler divergenciju**. Ako su  $P, Q$  dvije kategoričke raspodjele, pri čemu se  $P$  uzima kao ground-truth raspodjele, onda je:

$$KL(P \parallel Q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right)$$

gdje je  $\mathcal{X}$  zajednički prostor ishoda za  $P, Q$ . Uzima vrijednosti iz

intervala  $[0, +\infty)$ , i definisana je samo ako  $q(x) = 0$  povlači  $p(x) = 0$ , inače se vrijednost postavlja na  $+\infty$ . Naravno, što je manja vrijednost, to znači da je raspodjela  $Q$  bliža raspodjeli  $P$ . U našem slučaju imali bi:

$$KL(p_t||p_s) = \sum_c p_t(c) \log p_t(c) - \sum_c p_t(c) \log(p_s(c))$$

Sumiranje vršimo po klasama. Prva suma je suštinski **entropija** raspodjele  $p_t$ , a pošto je model  $T$  fiksiran prvi sumu možemo ignorisati tokom optimizacije. Druga suma je **cross-entropy** između raspodjela  $p_t$  i  $p_s$ , i nju razmatramo tokom optimizacije. Pored  $KL(p_t||p_s)$  često loss funkcija uključuje i originalni zadatak koji rješava model  $S$  - za slučaj klasifikacije to je kategorički log-likelihood. Tipično se u ovom framework-u model  $T$  naziva **učiteljem**, a model  $S$  se naziva **studentom**. Postoji još jedan detalj, a to je skaliranje raspodjela  $p_t, p_s$  prilikom računanja cross-entropy dijela:

$$p_t \sim \text{softmax}\left(\frac{h_t}{T}\right)$$

$$p_s \sim \text{softmax}\left(\frac{h_s}{T}\right)$$

$h_t, h_s$  su ne-normalizovane vjerovatnoće koje dobijamo iz modela  $T, S$  redom.  $T > 0$  je parametar **temperature**, i on kontroliše modalnost raspodjela  $p_t, p_s$ . Za  $T = 1$  imamo normalni softmax, za  $T > 1$  smanjuje se pouzdanost predikcija oba modela, dok za  $T < 1$  se povećava pouzdanost predikcija za oba modela. Tipično se u praksi stavlja  $T > 1$  kako bi obeshrabrili studentski model da se potpuno prilagođava učiteljskom modelu.

## 4 Rezultati

Kao što smo već nagovijestili, isprobaćemo 3 modela:

- **Sentence Transformer** - all-distilroberta-v1 Manji RoBERTa model od  $\approx 82M$  parametara dobijen primjenom knowledge distillation tehnike iz RoBERTa-base modela sa  $\approx 125M$  parametara, a zatim fine-tuneovan za semantic text similarity problem, slično kao što smo opisali u sekciji 2.2.3. Nije vršen dalji fine-tuning ovog modela (iako je moguć koristeći sentence-transformers biblioteku), poenta je bila da vidimo koliko jedan model optimizovan za retrieval može biti korisan za klasifikaciju. Kroz naše eksperimente primijetili smo da je ključni hiperparametar labele koje koristimo za klasifikaciju. Svaku rečenicu/komentar enkodiramo u vektor, a zatim računamo skalarni proizvod između dobijenog vektora i vektora za labele koje predstavljaju da li je neka rečenica govor mržnje ili ne. Eksperiment smo sproveli nad cijelim dataset-om, nismo koristili validation split jer ovdje ne sprovodimo nikakav fine-tuning. Rezultati se mogu vidjeti u tabeli 1. Prvo, nismo očekivali da će model biti ovoliko osjetljiv na izabrane labele, čak i onda kada se da deskriptivni opis jedne rečenice koja predstavlja govor mržnje, treća instanca u tabeli 1. Najbolji  $F_1$  dala je druga instanca, 0.71. Ima dosta mjesta za poboljšanja, tako da zaključujemo da za ovaj dataset nije od velike koristi izabrani retrieval model. Vjerovatno bi se preformance poboljšale ako bi se sproveo fine-tuning, ali opet naglašavamo da je poenta bila da se iskoristi **of-the-shelf** retrieval model za klasifikaciju.

Labela za govor mržnje	Labela za govor koji nije govor mržnje	$F_1$
This is hateful speech.	This is not hateful speech	0.64
<b>This is hateful and toxic content</b>	<b>This is not hateful and non toxic content</b>	<b>0.71</b>
Hatred, profanity, racial slurs.	Friendly, pleasantry, civility, gesture, levity.	0.63

Tabela 1: Rezultati STS eksperimenta.

- **Enkoder transformer** - roberta-base uz dodavanje LoRA slojeva ranga  $r = 8$ . Polazni model ima  $\approx 125M$  parametara, dok odabrana LoRA varijanta ima  $\approx 1.9M$  što je 65 puta manje parametara za LoRA verziju. Trening je sproveden u 8 epoha, za šta je na grafičkoj kartici **NVIDIA RTX 3060** sa 12 GB VRAM memorije trebalo približno 26 sati. Koristili smo sledeće hiperparametere:

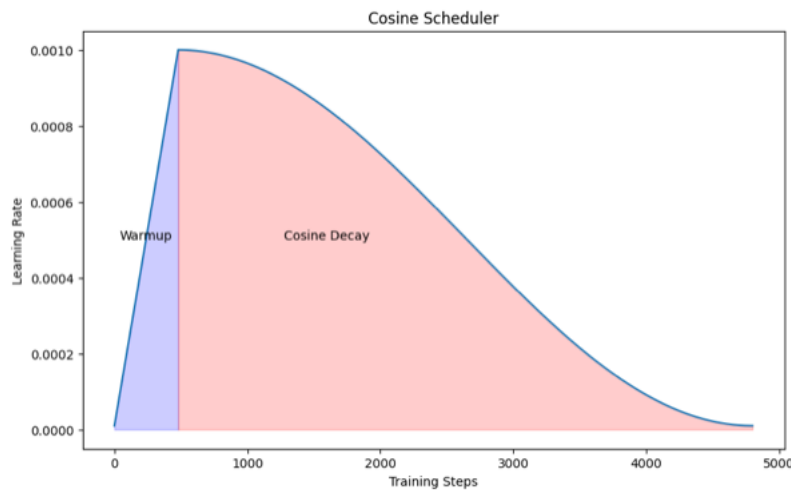
- Dropout klasifikator sloja: 0.1
- Dropout LoRA slojeva: 0.1
- LoRA  $\alpha = 0.8$
- Learning rate  $\lambda = 10^{-5}$
- Weight decay  $w = 10^{-3}$
- Batch size 8
- Cosine learning rate scheduler, pri čemu je warmup-rate 0.1. Cosine learning rate scheduling sprovodi se po formuli:

$$\lambda(s) = \begin{cases} \lambda(0) \frac{s}{W}, & s \leq W \\ \frac{\lambda(0)}{2} (1 + \cos(\pi \frac{s-W}{N-W})) & \end{cases}$$

$s$  je broj trenutne iteracije,  $N$  je totalni broj iteracija za zadatak,  $W$  je broj iteracija za koje se vrši warmup. U našem slučaju on je implicitno određen sa warmup rate parametrom - u prvih 10% iteracije se vrši warmup. Ilustracija kako se ponaša learning rate pri ovom rasporedu može se vidjeti na slici 24.

- AdamW algoritam za optimizaciju prvog reda, uz podrazumijevane vrijednosti  $\beta_1 = 0.9, \beta_w = 0.999$

Rezultujući model ostvario je **0.9**  $F_1$  na dobijenom validation skupu, što je značajno poboljšanje u odnosu na prethodni metod. Već smo pričali o zakonima skaliranja za transformer neuronske



Slika 24: Ilustracija cosine learning rate with warmup algoritma. Izvor: [link](#)

mreže, tako da očekujemo da bi sa povećanjem LoRA ranga dobili model sa još boljim performansama. Još jedan pravac poboljšanja jeste opreznija kalibracija hiperparametara, jedna popularna opcija u praksi je Optuna Python biblioteka koja gotovo automatizuje ovaj proces koristeći algoritme iz Bayesian optimization familije.

- **Enkoder-dekoder transformer** - google/flan-t5-base instanca prethodno opisan FLAN T5 modela sa  $\approx 248M$  parametara. Ovdje smo koristili LoRA rang 18, primjetno veći u odnosu na rang 8 koji smo imali kod RoBERTa modela, zato što smo LoRA kod RoBERTa modela podesili za sve linearne transformacije, a ovdje targetiramo samo query i key linearne transformacije unutar attention blokova, čime postizemo da LoRA varijanta ovog modela ima  $\approx 1.9M$  parametara. Primijetimo da je ovaj broj približno isti kao broj parametara LoRA instance RoBERTa modela, namjerno smo skalirali modele da imaju isti broj parametara kako bi i po tom kriterijumu vršili poređenje. Što se tiče ostalih hiperparametara, većina je zadržala istu vrijednost kao kod RoBERTa instance, navodimo samo one hiperparametre čija



se vrijednost razlikuje:

- Learning rate  $\lambda = 3 \times 10^{-4}$ . Autori [2] članka su za fine-tuning koristili odprilike ovaj learning rate, veći nego kod RoBERTa modela, tako da smo se i mi pridržavali sličnih vrijednosti.
- Batch size 16, implementiran kao batch size 8 uz akumulaciju gradijenata u 2 koraka zbog memorijskih nedostataka. Takođe su autori [2] koristili i veći batch size u odnosu na RoBERTa, tako da se i mi pridržavamo ovog pravila.
- Labele za autoregresivno dekodiranje: **Hateful** za rečenice koje su govor mržnje, **Non-hateful** za rečenice koje nisu govor mržnje.
- Prefiks za ulazne rečenice: **Classify this sentence as hateful or non-hateful speech**. Tokom treninga, enkoder komponenti predajemo **Classify this sentence as hateful or non-hateful: < rečenica >**, a dekodeer obučavamo tako da na izlazu da odgovarajuću labelu, **Hateful** ili **Non-hateful**.

Trening je pokrenut sa 8 epoha na istom hardveru kao i za RoBERTa, međutim već nakon 4 epohe uočili smo overfitting, tako da je trening nakon 4 epohe prekinut. Model je postigao **0.86**  $F_1$ , nešto lošiji od prethodno opisanog RoBERTa modela, što nagoviještava da je za problem klasifikacije pogodniji model koji ga riješava na "konvencionalni" način (modeliranje vjerovatnosne raspodjele klasa). Ipak ovo nije definitivni zaključak, vjerovatno bi se pažljivijim odabirom hiperparametara mogli poboljšati rezultati. Overfitting potencijalno ukazuje na veliki learning rate, tako da bi i to mogao da bude jedan pravac poboljšanja ovog modela. Na kraju, napominjemo da je za treniranje ovog modela kroz 4 epohe bilo potrebno  $\approx 43$  sata, što je značajno duže u odnosu na prethodni RoBERTa model - 8 epoha za  $\approx 26$  sati. Ovo drastično povećanje potiče uglavnom od dva

faktora:

- U poglavlju 1.5 izložili smo loss funkciju koju koristi dekodeoer transformer, enkoder-dekodeoer transformer koristi suštinski istu loss funkciju samo je izlaz dekodeoer komponente dodatno uslovljen izlazom enkoder komponente, znatno složenija nego kategorički log-likelihood za "konvencionalne" modele za klasifikaciju. Iako je broj parametara za instance RoBERTa i FLANrT5 modela približno isti, zbog složenije loss funkcije u drugom slučaju imamo da će backpropagation algoritmu u svakom koraku učenja trebati više vremena za računanje parcijalnih izvoda nego kod RoBERTa modela.
- Tokom evaluacije modela, nemamo računanje parcijalnih izvoda, ali se predikcija kod enkoder-dekodeoer modela vrši na značajno drugačiji način u odnosu na "konvencionalne" modele za klasifikaciju. Kod "konvencionalnih" modela za klasifikaciju jednostavno uzmemo najvjerovalniju klasu, kod enkoder-dekodeoer modela (isto važi za dekodeoer modele) figuriše nam i algoritam za dekodiranje koji koristimo. Kao što smo naveli u jednoj od prethodnih sekcija, u praksi je ovo najčešće **top-p + temperature softmax** algoritam. Kod **transformers** Python biblioteke koje smo koristili, tokom treniranja podrazumijevani algoritam je **greedy decoding** (link) tako da smo se i mi pridržavali istog metoda. Očekivano, autoregresivno dekodiranje je vremenski zahtjevnija operacija u odnosu na pronalaženje najveće vjerovalnoće, pa je i evaluacija enkoder-dekodeoer modela sporija. Iako se na algoritam za dekodiranje može gledati kao na još jedan hiperparametar, naše labele za klase su relativno kratke, tokenizacija od **Hateful** se svodi na svega nekoliko tokena, pa ne očekujemo da bi promjena algoritma za dekodiranje značajno uticala na rezultate.

Najzad, rezultate prethodno opisanih eksperimenata dajemo u tabeli 2.

Model	Broj Param.	LoRA rang	LoRA broj param.	Broj epoha	Trening	$F_1$
all-distilroberta-v1	82M	–	–	–	–	0.71
google/flan-t5-base	248M	18	1.9M (130x manje)	4	43 h	0.86
roberta-base	125M	8	1.9M (65x manje)	8	26 h	0.9

Tabela 2: Pregled rezultata.

## Bibliografija

- [1] Robin Brochier, Adrien Guille, and Julien Velcin. “Global Vectors for Node Representations”. In: *CoRR* abs/1902.11004 (2019). arXiv: 1902.11004. URL: <http://arxiv.org/abs/1902.11004>.
- [2] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 2022. arXiv: 2210.11416 [cs.LG]. URL: <https://arxiv.org/abs/2210.11416>.
- [3] Alexis Conneau et al. “Unsupervised Cross-lingual Representation Learning at Scale”. In: *CoRR* abs/1911.02116 (2019). arXiv: 1911.02116. URL: <http://arxiv.org/abs/1911.02116>.
- [4] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: <http://arxiv.org/abs/1810.04805>.
- [5] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: <http://arxiv.org/abs/1512.03385>.
- [6] Edward J. Hu et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: <https://arxiv.org/abs/2106.09685>.
- [7] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361. URL: <https://arxiv.org/abs/2001.08361>.
- [8] Vladimir Karpukhin et al. “Dense Passage Retrieval for Open-Domain Question Answering”. In: *CoRR* abs/2004.04906 (2020). arXiv: 2004.04906. URL: <https://arxiv.org/abs/2004.04906>.
- [9] Taku Kudo and John Richardson. “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing”. In: *CoRR* abs/1808.06226 (2018). arXiv: 1808.06226. URL: <http://arxiv.org/abs/1808.06226>.
- [10] Jindrich Libovický, Rudolf Rosa, and Alexander Fraser. “How Language-Neutral is Multilingual BERT?” In: *CoRR* abs/1911.03310

- (2019). arXiv: 1911.03310. URL: <http://arxiv.org/abs/1911.03310>.
- [11] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pre-training Approach”. In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: <http://arxiv.org/abs/1907.11692>.
  - [12] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: <https://arxiv.org/abs/1301.3781>.
  - [13] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *CoRR* abs/2103.00020 (2021). arXiv: 2103.00020. URL: <https://arxiv.org/abs/2103.00020>.
  - [14] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. URL: <https://arxiv.org/abs/1910.10683>.
  - [15] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *CoRR* abs/1908.10084 (2019). arXiv: 1908.10084. URL: <http://arxiv.org/abs/1908.10084>.
  - [16] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *CoRR* abs/1508.07909 (2015). arXiv: 1508.07909. URL: <http://arxiv.org/abs/1508.07909>.
  - [17] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-Attention with Relative Position Representations”. In: *CoRR* abs/1803.02155 (2018). arXiv: 1803.02155. URL: <http://arxiv.org/abs/1803.02155>.
  - [18] Xinying Song et al. “Linear-Time WordPiece Tokenization”. In: *CoRR* abs/2012.15524 (2020). arXiv: 2012.15524. URL: <https://arxiv.org/abs/2012.15524>.
  - [19] Stanford. *CS 230 - Recurrent Neural Networks*. <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>.
  - [20] Matej Ulcar and Marko Robnik-Sikonja. “FinEst BERT and CroSlo-Engual BERT: less is more in multilingual models”. In: *CoRR*

- abs/2006.07890 (2020). arXiv: 2006 . 07890. URL: <https://arxiv.org/abs/2006.07890>.
- [21] Princeton University. *WordNet*. <https://wordnet.princeton.edu/>.
- [22] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: <http://arxiv.org/abs/1706.03762>.